사이버보안개론

Introduction to Cybersecurity

유준수 (Joon Soo Yoo, Ph.D.)

Fall 2026

Defense & Advanced Security Lab (DAS Lab)
Department of Advanced Defense Technology and Industry
Jeonbuk National University

Chapter 6. Malicious Software

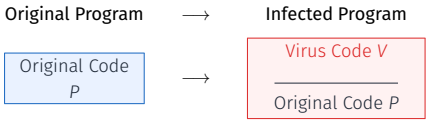
6.3 Propagation – Infected Content – Viruses

Virus Infection, Replication, and Evasion Techniques

1-1. Virus as Infected Executable Content

앞에서는 Virus가 다른 File에 자신을 복제한다는 개념을 살펴보았다.

이번에는 **어떻게 정상 Program을 감염시키고, 실행 흐름을 변경하는지** 살펴본다.



개념적으로 감염된 Program은 다음과 같이 표현할 수 있다.

$$P' = V + P$$

그러나 중요한 것은 단순히 Code를 추가하는 것이 아니다.

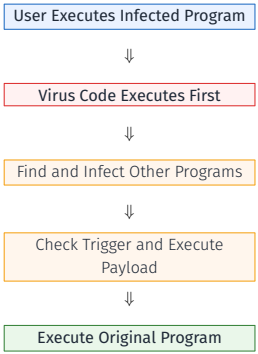
핵심

감염된 Program의 **Entry Point 또는 실행 흐름**이 Virus Code를 먼저 실행하도록 변경되어야 한다.

※ 실제 실행 파일은 Header, Section, Entry Point 등을 가지므로 단순한 Byte 연결만으로 감염이 완성되는 것은 아니다.

1-2. Executable Virus: Execution Flow

교재 Figure 6.1(a)는 가장 단순한 형태의 Executable Virus를 설명한다.



Why Execute the Original Program?

Virus가 자신의 작업을 마친 뒤 원래 Program을 실행하면, 사용자는 감염 사실을 알아차리기 어려울 수 있다.

1-3. Simple Virus: Pseudocode

교재 Figure 6.1(a)의 핵심 Logic을 단순화하면 다음과 같다.

```
procedure virus_main():

    target = find_uninfected_program()

    if target exists:
        mark_target_as_infected(target)

    if trigger_condition():
        execute_payload()

    execute_original_program()
```

Understanding the Logic

Function	Role
find_uninfected_program()	감염되지 않은 Program 탐색
mark_target_as_infected()	감염 과정의 개념적 표현
trigger_condition()	Payload 실행 조건 확인
execute_payload()	Virus의 추가 기능 수행
execute_original_program()	원래 Program 실행

※ 위 코드는 Figure 6.1의 동작을 설명하는 Pseudocode이며, 실제 파일 감염이나 자기 복제 기능을 구현하지 않는다.

1-4. Virus Infection and Access Control

Virus가 다른 Program을 감염시키려면 해당 Program의 내용을 수정할 수 있어야 한다.

Example: File Permissions

Target	Read	Write	Infection
User-owned Program	Yes	Yes	수정 가능성이 있음
System Program	Yes	No	일반 사용자 권한으로 수정 불가
Protected Executable	Yes	No	접근통제로 감염 제한

예를 들어 일반 사용자 권한으로 실행된 Virus가 관리자만 수정할 수 있는 System Program을 변경하려고 한다면 어떻게 될까?

Virus Execution $\longrightarrow$ **Write Attempt** $\longrightarrow$ Access Denied

Security Principle

File Permission과 Access Control은 전통적인 File-Infector Virus의 전파를 제한한다. 단, 사용자가 수정할 수 있는 File이나 권한이 탈취된 환경에서는 감염이 가능할 수 있다.

1-5. Compression Virus: Why Compression?

단순한 Virus는 정상 Program에 Virus Code를 추가하기 때문에 File Size가 증가할 수 있다.

Simple Virus

$$100 \text{ KB (Program)} + 10 \text{ KB (Virus)} = 110 \text{ KB}$$

공격자는 이러한 변화를 숨기기 위해 **Compression Virus**를 사용할 수 있다.

Original Program: 100 KB

⇓ Compress

Compressed Program: 90 KB

⇓ Add Virus Code

Virus: 10 KB + Compressed Program: 90 KB

$$\boxed{10 \text{ KB} + 90 \text{ KB} = 100 \text{ KB}}$$

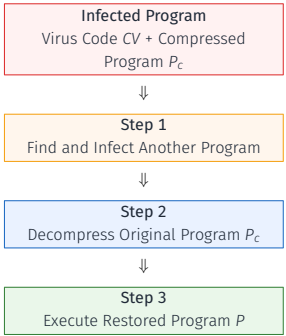
핵심

감염 전후 File Size가 같더라도 Program 내부의 Code는 변경되었을 수 있다.

※ 수치는 이해를 위한 예시이며, 모든 Program이 원하는 크기로 압축되는 것은 아니다.

1-6. Compression Virus: Execution Process

교재 Figure 6.2는 Compression Virus가 감염된 Program을 실행할 때의 과정을 보여준다.



Why Is This Effective?

Virus는 다른 Program을 감염시키면서도 원래 Program이 정상적으로 실행되도록 할 수 있다.
따라서 사용자가 감염 여부를 알아차리기 어려울 수 있다.

1-7. Compression Virus: Conceptual Code

Compression Virus의 핵심을 Pseudocode로 표현하면 다음과 같다.

```
procedure compression_virus():

    target = find_uninfected_program()

    if target exists:
        compressed = compress(target)
        conceptual_infection(target, compressed)

    original = decompress(embedded_program)

    execute(original)
```

Key Operations

Operation	Meaning
compress()	정상 Program을 압축
conceptual_infection()	압축된 Program과 Virus Code의 결합을 개념적으로 표현
decompress()	원래 Program을 복원
execute()	복원된 Program 실행

핵심

Compression Virus의 목적은 자기 복제 기능을 유지하면서 File Size 증가와 같은 감염 흔적을 줄이는 것이다.

1-8. Virus Classification: Infection Targets

Virus는 어떤 대상을 감염시키는지에 따라 분류할 수 있다.

Classification by Infection Target

Type	Target and Behavior
Boot Sector Infector	Boot Record 또는 MBR 감염
File Infector	Executable File 감염
Macro Virus	Document 내부의 Macro 또는 Script 감염
Multipartite Virus	여러 유형의 File 또는 System Area 감염

핵심

Virus의 종류는 Payload가 아니라 **어떤 대상을 어떻게 감염시키는가**에 따라 구분할 수도 있다.

※ Boot Sector Infector는 전통적인 BIOS/MBR 환경에서 특히 중요했던 유형이다. 현대 UEFI 환경의 Bootkit과는 구분하여 이해해야 한다.

1-9. Virus Classification: Concealment

Virus는 Security Software의 탐지를 피하기 위해 다양한 은폐 기법을 사용한다.

Classification by Concealment Strategy

Type	Technique
Encrypted Virus	Virus Code 일부를 Encryption하여 숨김
Stealth Virus	감염 흔적이나 존재 자체를 숨김
Polymorphic Virus	복제할 때마다 Code Pattern을 변경
Metamorphic Virus	자신의 Code 구조를 재작성

Signature-Based Detection

Known Byte Pattern $\longrightarrow$ Malware Detection

그러나 Virus가 복제될 때마다 Byte Pattern이 바뀐다면 단순한 Signature Matching이 어려워질 수 있다.

핵심

Virus의 Concealment Technique은 Malware의 기능뿐 아니라 탐지 가능한 Code Pattern 자체를 변경할 수 있다.

1-10. Polymorphic vs. Metamorphic Virus

Polymorphic Virus와 Metamorphic Virus는 모두 복제 과정에서 Code의 형태를 변경한다.
그러나 **변경하는 방식**이 다르다.

Comparison

	Polymorphic	Metamorphic
Main Idea	암호화 또는 Code 변형	Code 자체를 재작성
Code Pattern	복제마다 달라짐	복제마다 달라짐
Typical Method	Encryption, Mutation Engine	Instruction Rewriting
Goal	Signature Detection 회피	Signature Detection 회피

Polymorphic

$$E_{K_1}(V), \quad E_{K_2}(V), \quad E_{K_3}(V)$$

Metamorphic

$$V_1 \longrightarrow V_2 \longrightarrow V_3$$

핵심

Polymorphic은 주로 Encryption과 Mutation을 통해 서로 다른 표현을 만들고,
Metamorphic은 Virus Code 자체를 재작성한다.

1-11. (참고) Equivalent Code, Different Signature

같은 기능을 수행하는 Code라도 서로 다른 Instruction으로 표현할 수 있다.

Version A

```
int add_one(int x) {
    return x + 1;
}
```

Version B

```
int add_one(int x) {
    return x - (-1);
}
```

두 Code는 일반적인 정수 연산 관점에서 같은 결과를 계산한다.

그러나 Compiler와 Optimization 설정에 따라 생성되는 Machine Code가 같을 수도 있고 다를 수도 있다.

Different Code Representation

⇓

Potentially Different Byte Pattern

⇓

Signature Matching May Become Harder

핵심

Metamorphic Malware는 이와 같은 Code Transformation을 더 복잡하게 적용하여 서로 다른 형태의 Malware를 생성할 수 있다.

1-12. Key Takeaways

이번 절에서는 Virus의 종류보다 **내부 동작과 탐지 회피 원리**에 집중하였다.

Summary

Concept	Key Idea
Executable Virus	정상 Program에 Virus Code 삽입
Execution Flow	Virus Code 실행 후 원래 Program 실행
Access Control	다른 Program을 수정할 권한이 필요
Compression Virus	압축으로 File Size 증가를 숨길 수 있음
Polymorphic Virus	복제마다 Code Pattern 변경
Metamorphic Virus	Code 자체를 재작성

핵심

Virus는 단순히 악성 기능을 수행하는 Code가 아니라, 다른 실행 가능한 Content를 감염시키고 자신을 복제하는 Mechanism을 가진다.

Chapter 6. Malicious Software

6.4 Propagation – Vulnerability Exploit – Worms

Automated Propagation and Network Security

2-1. How Does a Worm Propagate?

앞에서는 Vulnerability를 이용하여 Malware가 실행되는 과정을 살펴보았다.

이번에는 **감염된 Computer가 어떻게 다른 Computer를 자동으로 감염시키는지** 살펴본다.



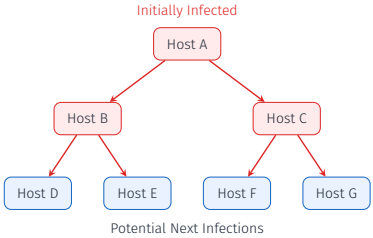
핵심

Worm은 감염된 Host가 다시 다른 Host를 감염시키는 **Automated Propagation Mechanism**을 가진다.

2-2. Worm Propagation: A Network Example

하나의 Computer가 감염되었다고 가정하자.

감염된 Computer는 다른 Host를 탐색하고, 새로운 감염 Host도 동일한 작업을 반복한다.



핵심

Worm의 위험성은 하나의 Host가 감염되는 것뿐 아니라, 감염된 Host가 새로운 공격의 출발점이 된다는 것이다.

※ 실제 전파는 취약점, 네트워크 연결성, 접근 권한 등에 따라 달라진다.

2-3. Target Discovery: Scanning Strategies

Worm은 새로운 Host를 감염시키기 전에 **어떤 Host를 공격할지** 결정해야 한다.

Target Discovery Strategies

Strategy	Description
Random Scanning	무작위 IP Address를 탐색
Hit-List Scanning	미리 수집한 Target List를 이용
Topological Scanning	감염된 Host가 가진 Network Information 활용
Local Subnet Scanning	감염된 Host와 동일한 Subnet의 다른 Host 탐색

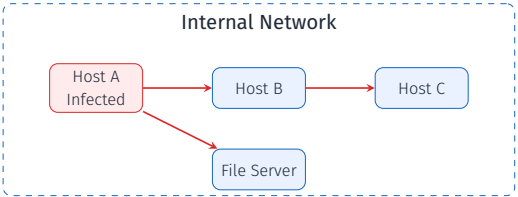
예를 들어 기업 내부 Computer가 감염되었다면, 공격자는 해당 Computer에서 접근 가능한 다른 Server나 Workstation을 발견할 수 있다.

핵심

Target Discovery는 단순히 IP Address를 찾는 것이 아니라, **공격 가능한 Host와 Service를 식별하는 과정**이다.

2-4. Local Subnet Propagation

외부 Firewall이 존재하더라도 내부 Computer가 감염되면 문제가 발생할 수 있다.



Internal Propagation

Security Implication

Perimeter Firewall만으로는 내부 Host 간 Malware Propagation을 충분히 방지하기 어렵다.
Network Segmentation과 Internal Access Control이 중요하다.

2-5. Worm Propagation Model

Worm은 감염된 Host가 다시 다른 Host를 감염시키므로 감염 규모가 빠르게 증가할 수 있다.

교재에서는 이를 Epidemic Model로 설명한다.

$$\boxed{\frac{dI(t)}{dt} = bI(t)S(t)}$$

Model Variables

Variable	Meaning
$I(t)$	Time t에서 감염된 Host 수
$S(t)$	감염되지 않은 취약한 Host 수
b	감염률을 결정하는 Parameter
N	전체 Host 수, $N = I(t) + S(t)$

Interpretation

감염된 Host가 많을수록 공격 시도도 증가하며, 감염 가능한 Host가 많을수록 새로운 감염이 발생할 가능성이 높아진다.

※ 감염과 취약 Host의 균일한 혼합을 가정한 단순화 모델이다.

2-6. Why Does Worm Propagation Accelerate?

초기에는 감염된 Host가 적지만, 새로 감염된 Host도 전파에 참여한다.

매 단계에서 각 Host가 새로운 Host 두 대를 감염시킨다고 가정하자.

Simplified Propagation Example

Round	Newly Infected	Total Infected	Growth
0	1	1	2^0
1	2	3	2^1
2	4	7	2^2
3	8	15	2^3
4	16	31	2^4
5	32	63	2^5

새로운 감염 수는

$$I_{\text{new}}(k) = 2^k$$

누적 감염 수는

$$I_{\text{total}}(k) = 2^{k+1} - 1$$

핵심

새로 감염된 Host가 다시 전파에 참여하기 때문에 초기에는 Exponential Growth가 나타날 수 있다.

※ 모든 Host가 매 Round 정확히 두 대를 감염시킨다는 가상의 예시이다.

2-7. Why Does Propagation Eventually Slow Down?

Worm이 무한히 Exponential Growth를 유지하는 것은 아니다.

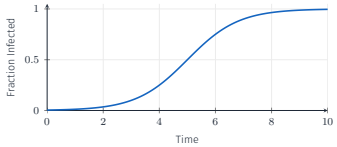
감염 가능한 Host 수에는 한계가 있기 때문이다.

$$N = I(t) + S(t)$$

따라서

$$\frac{dI}{dt} = bI(N - I)$$

이 식은 Logistic Growth 형태를 가진다.



핵심

초기에는 감염 속도가 증가하지만, 취약한 Host가 감소하면서 새로운 감염 속도는 점차 감소한다.

2-8. (참고) SQL Slammer: Rapid Propagation

2003년 SQL Slammer는 Microsoft SQL Server 관련 서비스의 Buffer Overflow Vulnerability를 악용하였다.

SQL Slammer

Concept	Description
Target	Microsoft SQL Server 관련 서비스
Vulnerability	Buffer Overflow
Propagation	UDP 기반 자동 전파
Speed	약 10분 만에 취약한 Host의 약 90% 감염
Impact	대규모 Network Congestion

SQL Slammer의 핵심은 복잡한 Payload가 아니라 **매우 빠른 자동 전파 자체가 Network Availability를 침해했다는 점**이다.

핵심

Worm은 별도의 파괴적인 Payload가 없어도 대량의 Network Traffic을 발생시켜 서비스 장애를 유발할 수 있다.

2-9. Modern Worm Propagation

현대적인 Worm은 하나의 취약점이나 하나의 Operating System에만 의존하지 않을 수 있다.

Modern Propagation Techniques

Technique	Description
Multi-Exploit	여러 Vulnerability를 조합하여 전파
Credential Abuse	탈취한 Credential 또는 약한 Password 활용
Cross-Platform	Windows, Linux, IoT 등 다양한 환경 공격
Internal Propagation	내부 Network의 Host와 Service 탐색
Payload Delivery	Ransomware, Botnet, Cryptominer 등 실행

핵심

Worm의 본질은 특정 Exploit이 아니라 **새로운 Host를 자동으로 발견하고 감염을 반복하는 Mechanism**이다.

※ Credential Abuse나 Remote Execution은 공격자가 수동으로 수행할 수도 있다. 자동 전파 여부가 Worm과 일반적인 Lateral Movement를 구분하는 중요한 기준이다.

2-10. Detecting and Containing Worm Propagation

Worm의 자동 전파는 Network와 Host에 관찰 가능한 흔적을 남길 수 있다.

Detection and Defense

Mechanism	Security Countermeasure
Repeated Scanning	Network IDS, Traffic Monitoring
Vulnerability Exploitation	Security Patch, Vulnerability Management
Unauthorized Execution	EDR, Application Control
Internal Propagation	Network Segmentation, Internal Firewall
Credential Abuse	MFA, Least Privilege, Credential Protection

Early Detection $\longrightarrow$ **Host Isolation** $\longrightarrow$ **Propagation Containment**

핵심

Worm Defense의 목표는 최초 감염을 방지하는 것뿐 아니라, **감염된 Host가 다른 Host를 공격하지 못하도록 전파 경로를 차단하는 것**이다.

2-11. Key Takeaways

이번 절에서는 Worm의 자동 전파 원리와 그에 따른 Security Implication을 살펴보았다.

Summary

Concept	Key Idea
Worm Propagation	감염된 Host가 다른 Host를 자동 감염
Target Discovery	공격 가능한 Host와 Service 탐색
Scanning Strategies	Random, Hit-List, Topological, Local Subnet
Propagation Model	감염된 Host와 취약한 Host 수에 따라 확산
Rapid Propagation	초기 감염 규모가 빠르게 증가할 수 있음
Containment	Early Detection, Host Isolation, Segmentation

핵심

Worm은 하나의 Vulnerability가 Network 전체의 Security Incident로 확대될 수 있음을 보여준다.

Chapter 6. Malicious Software

6.5–6.9 Malware Propagation and Payloads

Social Engineering, System Corruption, and Stealth

3-1. Social Engineering and Trojan Horses

Malware는 Software Vulnerability뿐 아니라 **사용자의 판단과 행동**을 이용하여 전파될 수 있다.

Social Engineering and Malware

Concept	Description
Spam E-Mail	불특정 다수에게 대량으로 전송되는 E-Mail
Phishing	신뢰할 수 있는 기관이나 사람을 사칭하여 Credential 또는 개인정보 탈취
Spear Phishing	특정 개인이나 조직에 맞춘 Targeted Phishing
Trojan Horse	정상적인 Software로 위장하여 사용자가 악성 기능을 실행하도록 유도

Example

Fake Security Update

⇓

User Downloads and Executes

⇓

Malware Installation

핵심

Social Engineering은 사용자를 속이는 공격 방식이고, Trojan은 정상적인 Software처럼 보이도록 위장한 Malware이다.
두 개념은 서로 결합되어 사용될 수 있다.

3-2. Malware Payload: System Corruption

Malware가 Target System에서 실행되면 Propagation 이외에 다양한 Payload를 수행할 수 있다.

System Corruption Payloads

Payload	Security Impact
Data Destruction	File 또는 System Data 삭제 및 손상
Ransomware	File Encryption을 통한 접근 제한 및 금전 요구
Wiper	Data를 삭제하거나 손상시켜 복구를 어렵게 만듦
Logic Bomb	특정 Trigger Condition이 만족되면 악성 기능 실행
Physical Damage	제어 시스템을 조작하여 물리적 장비에 피해 유발

Propagation $\neq$ Payload

How Malware Spreads $\neq$ What Malware Does

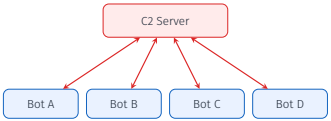
핵심

Propagation은 Malware가 확산하는 방법이고, Payload는 감염 이후 수행하는 악성 기능이다.

3-3. Botnet and Command-and-Control

Bot은 공격자의 명령에 따라 동작하는 Compromised Computer 또는 Malware Agent이다.

여러 Bot을 연결하여 제어하는 구조를 **Botnet**이라고 한다.



Common Botnet Activities

Activity	Description
DDoS	다수의 Bot을 이용한 Service Disruption
Spam	대량의 악성 E-Mail 전송
Information Theft	Credential 및 Sensitive Data 수집
Payload Delivery	추가 Malware 다운로드 및 실행

핵심

Worm은 자동 전파가 핵심이고, Botnet은 감염된 System에 대한 **원격 제어와 협력적 공격**이 핵심이다.

3-4. (참고) How Does a Bot Receive Commands?

Bot은 공격자의 C2 Server와 통신하여 새로운 명령을 확인할 수 있다.

```pseudocode
# Conceptual pseudocode

while active:

    command = check_c2_server()

    if command == "status":
        report_status()

    elif command == "update":
        request_update()

    wait_for_next_check()
```

Command-and-Control Process

Step	Description
Check-in	Bot이 C2 Server에 접속
Command Retrieval	새로운 명령 확인
Command Execution	전달된 명령에 따른 동작 수행
Reporting	실행 결과를 C2 Server에 전달

핵심

Botnet은 C2를 통해 여러 감염된 Host를 제어할 수 있다.
C2 Communication은 HTTP(S), DNS, Peer-to-Peer 등 다양한 방식으로 구현될 수 있다.

※ 위 코드는 C2의 동작 구조를 설명하는 Pseudocode이며, 실제 네트워크 통신이나 원격 명령 실행 기능은 없다.

3-5. Information Theft and Data Exfiltration

Malware는 System을 손상시키는 것뿐 아니라 사용자의 Sensitive Information을 탈취할 수도 있다.

Information Theft Techniques

Technique	Description
Keylogger	사용자의 Keyboard Input 수집
Spyware	사용자의 활동 및 Sensitive Information 감시
Credential Theft	Password, Token 등 인증정보 탈취
Data Exfiltration	탈취한 정보를 외부 System으로 전송

Malware Execution

⇓

Sensitive Data Collection

⇓

Data Exfiltration

⇓

Confidentiality Violation

핵심

Information Theft Malware는 주로 Confidentiality를 침해한다.
특히 Credential이 탈취되면 추가적인 Unauthorized Access로 이어질 수 있다.

3-6. Backdoors and Rootkits

Backdoor와 Rootkit은 모두 Malware가 System 내부에서 활동하는 데 사용될 수 있다.
그러나 두 기술의 목적은 다르다.

Backdoor vs. Rootkit

Technique	Main Purpose
Backdoor	정상적인 Authentication 또는 Access Control을 우회하는 비인가 접근 경로 제공
Rootkit	Malware의 존재나 System 변경 사항을 사용자와 Security Software로부터 은폐

Backdoor

Unauthorized Access

Rootkit

Concealment of Malicious Activity

핵심

Backdoor는 **접근 경로를 제공**하고, Rootkit은 **존재를 숨기는 것**에 초점을 둔다.
두 기술은 함께 사용될 수도 있다.

3-7. (참고) How Does a Rootkit Hide a Process?

Rootkit은 System Information을 반환하는 경로를 조작하여 Malware의 존재를 숨길 수 있다.

Actual Process List

```
PID 100  explorer
PID 200  browser
PID 300  malware
```

Manipulated Result

```
PID 100  explorer
PID 200  browser
```

User Program

⇓

System Call / API

⇓

Rootkit Interception

⇓

Manipulated Process Information

핵심

Rootkit은 Malware Process를 실제로 제거하지 않고도 Process Listing 결과에서 숨길 수 있다.
따라서 Security Tool이 관찰하는 정보 자체가 조작될 수 있다.

※ System Call Hooking은 Rootkit 은폐 원리를 설명하는 대표적인 역사적 기법이다. 현대 OS에서는 Kernel Integrity Protection 등으로 이러한 조작을 제한한다.

Chapter 6. Malicious Software

6.10 Countermeasures

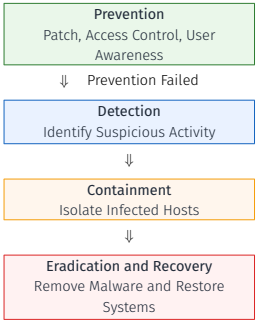
Prevention, Detection, and Response

4-1. Malware Countermeasures: Overall Strategy

Malware에 대한 가장 이상적인 대응은 **감염 자체를 예방하는 것(Prevention)**이다.

그러나 모든 Malware Infection을 예방하는 것은 어렵다.

따라서 감염 이후의 Detection과 Response도 필요하다.



핵심

Malware Defense는 단일 Security Product가 아니라 Prevention, Detection, Response, Recovery를 결합한 **Defense-in-Depth Strategy**이다.

4-2. Prevention: Reducing the Attack Surface

Malware가 침입하거나 확산할 수 있는 경로를 줄이는 것이 Prevention의 핵심이다.

Malware Prevention

Countermeasure	Purpose
Security Patch	알려진 Vulnerability를 수정하여 Exploit 가능성을 감소
Access Control	사용자와 Program의 권한을 제한
Least Privilege	필요한 최소 권한만 부여하여 Malware의 피해 범위 제한
Network Segmentation	감염된 Host의 내부 Network 확산 제한
User Awareness	Phishing, Malicious Attachment 등의 Social Engineering 공격 예방
Application Control	허용되지 않은 Program의 실행 제한

핵심

Prevention은 Malware를 탐지하기 전에 **공격이 성공할 수 있는 조건 자체를 줄이는 것**이다.

4-3. Malware Detection Approaches

Malware Detection은 크게 Signature, Heuristic, Behavior 기반으로 구분할 수 있다.

Detection Techniques

Method	Approach	Limitation
Signature-Based	알려진 Malware의 Pattern과 비교	Unknown Malware 탐지 한계
Heuristic-Based	의심스러운 Code Structure와 특징 분석	False Positive 가능성
Behavior-Based	실행 중 File, Process, Network Activity 관찰	실행 이후 탐지될 수 있음

Signature

What Does the Code Look Like?

Behavior

What Does the Program Actually Do?

핵심

Modern Malware Detection은 하나의 방식만 사용하지 않고 여러 Detection Technique을 결합한다.

4-4. (참고) Generic Decryption and Emulation

Polymorphic Malware는 Code를 암호화하여 Signature-Based Detection을 어렵게 만들 수 있다.

$$V = D + E_K(M)$$

- D: Decryption Routine
- $E_K(M)$: Encrypted Malware Code
- M: Original Malware Code

Anti-Virus는 안전하게 통제된 Emulation Environment에서 Code를 실행하여 복호화 과정을 관찰할 수 있다.

Encrypted Malware

⇓

Emulation / Decryption

⇓

Decrypted Code

⇓

Malware Analysis

핵심

Malware가 실행 과정에서 자신의 Code를 복호화한다면, Emulator는 복호화된 Code를 분석할 수 있다.
이는 현대 Sandbox Analysis와 연결되는 개념이다.

4-5. Endpoint Detection and Response (EDR)

EDR (Endpoint Detection and Response)은 Endpoint에서 발생하는 Security Event를 수집하고, 위협을 탐지하고 대응하는 기술이다.

Example: Endpoint Events

Time	Observed Event
10:00:01	Unknown Process Executed
10:00:02	Process Opened 500 Files
10:00:03	Process Modified 300 Files
10:00:04	Suspicious Network Connection

각 Event는 개별적으로 정상적인 행동일 수 있다.

그러나 여러 Event가 연속적으로 발생한다면 Malware Activity를 의심할 수 있다.

Event Collection ⟶ Correlation ⟶ Detection ⟶ Response

핵심

EDR은 단순히 Malware File을 찾는 것을 넘어 **Process와 System Behavior를 분석**하여 공격을 탐지하고 대응하는 데 활용된다.

4-6. Rootkit Detection and Recovery

Rootkit은 Operating System이 제공하는 System Information을 조작할 수 있다.

따라서 Security Tool이 관찰한 결과를 항상 신뢰할 수 있는 것은 아니다.

Example: Inconsistent System Views

Observation	Process List
Normal OS API	Process A, Process B
Independent Trusted View	Process A, Process B, Process C

두 결과가 다르다면 System Information Manipulation을 의심할 수 있다.

Rootkit Countermeasures

Technique	Purpose
Secure Boot	신뢰된 Boot Component 실행 검증
Kernel Integrity Protection	Kernel Code 및 중요 구조 보호
Trusted Offline Analysis	감염된 OS를 신뢰하지 않고 분석
System Reinstallation	신뢰 가능한 Image로 System 재구축

핵심

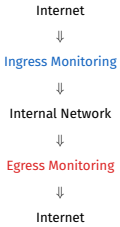
Kernel-Level Rootkit이 System의 신뢰성을 훼손했다면, Malware File 삭제만으로 안전한 복구를 보장하기 어렵다.

4-7. Network Monitoring: Ingress vs. Egress

Network-Based Detection은 Host 외부에서 Malware Traffic을 관찰한다.

Ingress and Egress Monitoring

Type	Direction	Example
Ingress	External → Internal	외부의 취약점 공격 시도
Egress	Internal → External	감염 Host의 C2 Communication



핵심

Ingress Monitoring은 외부에서 들어오는 공격을, Egress Monitoring은 내부에서 외부로 나가는 의심스러운 Traffic을 탐지하는 데 활용된다.
내부 Host 간 전파를 관찰하려면 Internal Network Monitoring도 필요하다.

4-8. SIEM and XDR

여러 Security Sensor에서 발생하는 Event를 통합하면 개별 Sensor가 발견하지 못한 Attack Pattern을 식별할 수 있다.

Security Monitoring Platforms

Technology	Main Purpose
SIEM	Security Information and Event Management 여러 System의 Log와 Security Event를 수집, 분석, 상관관계 분석
XDR	Extended Detection and Response Endpoint, Network, Cloud 등 여러 보안 영역의 탐지와 대응을 통합

Endpoint Logs + Network IDS + DNS Logs + Firewall Logs

⇓

Centralized Security Analysis

⇓

Threat Detection and Response

핵심

SIEM은 광범위한 Log 및 Event 분석에, XDR은 여러 보안 영역의 Threat Detection과 Response 통합에 초점을 둔다.
두 기술은 기능이 일부 중복되지만 동일하지 않다.

4-9. Defense in Depth

어떤 Security Mechanism도 모든 Malware Attack을 완벽하게 방어할 수 없다.

따라서 여러 Security Layer를 결합해야 한다.

Defense-in-Depth Architecture

Layer	Security Mechanism
User	Security Awareness, Phishing Prevention
Application	Application Control, Security Updates
Endpoint	Anti-Virus, EDR, Least Privilege
Network	Firewall, IDS/IPS, Segmentation
Organization	SIEM, XDR, Incident Response
Recovery	Backup, System Restoration

Prevent → **Detect** → **Contain** → **Recover**

핵심

Defense in Depth는 하나의 방어가 실패하더라도 다른 Security Layer가 공격의 성공이나 피해 확산을 제한하도록 설계하는 전략이다.

4-10. Chapter 6: Key Takeaways

Chapter 6에서는 Malware의 종류, Propagation, Payload, Countermeasures를 살펴보았다.

Chapter Summary

Concept	Key Question
Malware Type	What Kind of Malware Is It?
Propagation	How Does Malware Spread?
Vulnerability Exploitation	How Does Malware Gain Execution?
Payload	What Does Malware Do After Infection?
Stealth	How Does Malware Hide Its Presence?
Countermeasures	How Can We Prevent, Detect, and Respond?

Final Takeaway

Malware Security는 악성코드의 이름을 암기하는 것이 아니라, **공격이 어떻게 시작되고, 확산되고, 피해를 발생시키는지** 이해하는 것이다.
이를 바탕으로 적절한 Prevention, Detection, Response Mechanism을 설계해야 한다.

Thank You

Questions?