

# Malicious Software

---

UT CS361S

FALL 2020

LECTURE NOTES

A solid orange horizontal bar at the bottom of the slide.

# What is “Malicious Code”?

---

“Software or firmware intended to perform an ***unauthorized*** process that will have ***adverse impacts*** on the confidentiality, integrity, or availability of a system. A virus, worm, Trojan horse, or other code-based entity that infects a host. Spyware and some forms of adware are also examples of malicious code.”

NIST Special Publication 800-53, Revision 5

# Common Malware Classes

---

Malware can be classified by how it spreads or generically behaves

- Virus – typically has to be attached to another program (infection)
- Worm – typically spreads via network vulnerabilities
- Trojan Horse – typically appears benign but contains hostile operations

Malware can also be classified by its behavior

- Spyware – typically designed to steal information, observe behavior, etc.
- Adware – typically designed to “trap” a user into viewing certain ads
- Ransomware – typically locks data unless the user pays a ransom

# Two Primary Components

---

Payload – The code that performs the (harmful) action

Attack Vector/Exploit/Delivery – The code that enables the payload

- May include a transmission component
- May include a stealth component
- May include a mechanism for bypassing security
- May be as simple as an email with an attachment

# Malware History

---

Much of early computer security driven by military concerns

The biggest concern was an unauthorized user or program

Other concerns developed over time

The following slides are a very brief overview/highlight

# 1972 Government Report:

---

The technical issue of multilevel computer security is concerned with the concept of malicious threat. By this we recognize that the nature of shared use multilevel computer systems present to a malicious user a unique opportunity for attempting to subvert through programming the mechanism upon which security depends (i. e. , the control of the computer vested in the operating system). This threat, coupled with the concentration of the application (data, control system, etc. ) in one place (the computer system) makes computers a uniquely attractive target for malicious (hostile) action. Recognition of the implication of malicious threat is important to understanding the security limitations surrounding application of contemporary computer systems. The threat that a single user of a system operating as a hostile agent can simply modify an operating system to by-pass or suspend security controls, and the fact that the operating system controlling the computer application(s) is developed outside of USAF control, contribute strongly to the reluctance to certify (i. e. , be convinced) that contemporary systems are secure or even can be secured.

<https://apps.dtic.mil/sti/pdfs/AD0758206.pdf>

# 1987: Fred Cohen's Viruses

---

“Computer Viruses: Theory and Experiments” Fred Cohen, 1987

Introduced the concept of a self-replicating, evil program

The program attaches to a “good” program infecting it

When the infected program is run, the virus runs

The virus does it's evil AND spreads itself to other programs

Concepts first proposed by Cohen in 1984

# 1988: Morris Worm

---

Robert Morris wrote a self-spreading piece of code (worm)

Spread using exploits in:

- send mail,
- Finger
- rsh/rexec

Also guessed weak passwords

Copied code to new machine, compiled, and executed

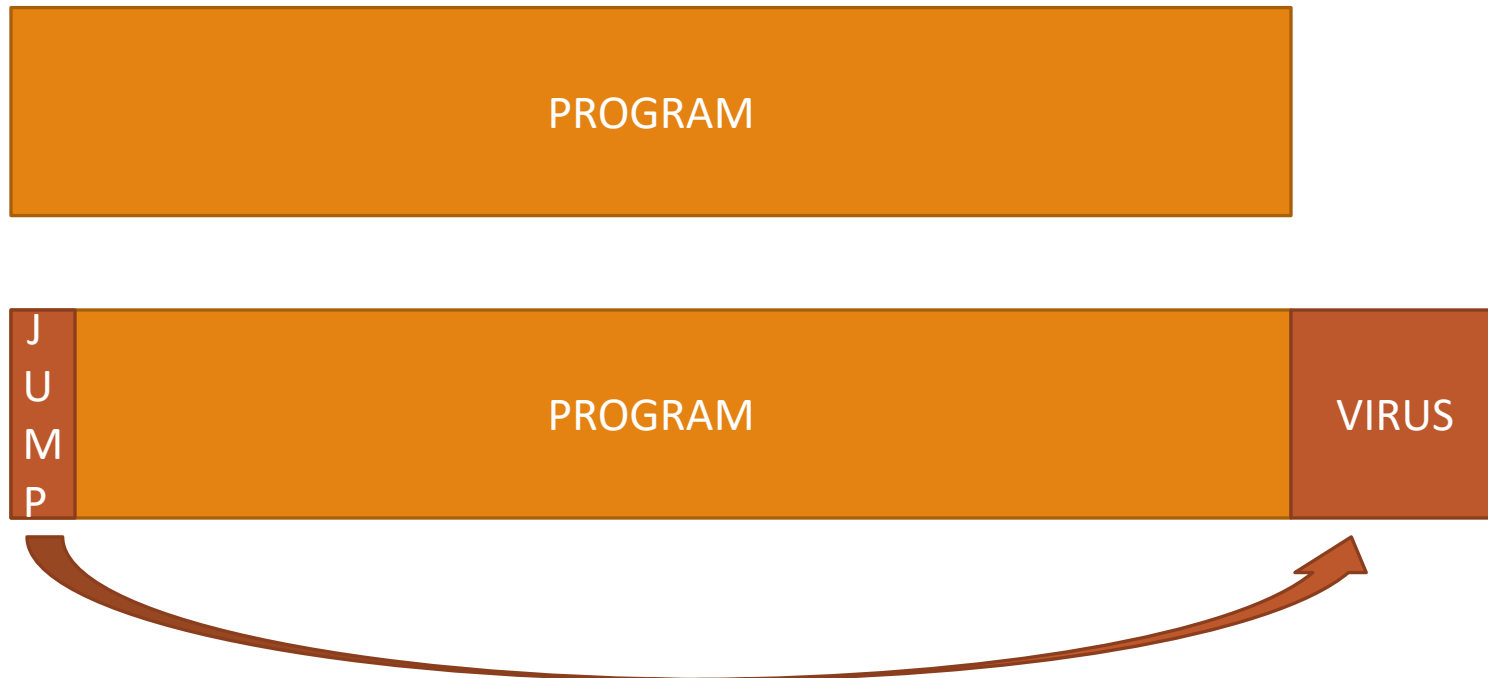
Accidentally re-infected machines until machines became unusable

DOS attack brought down the Internet



# Early DOS Viruses

---



# Impact of Early Viruses

---

Amazingly, most early malware ***DID MINIMAL DAMAGE***

Often just a delivery system with a weak payload

Many viruses spread for the sake of spreading

Even the Morris worm was disruptive by accident

There were exceptions, but also plenty of hype

# 1995: Concept Macro Virus

---

Microsoft Word and Excel have limited scripting (“Macros”)

Concept was the first virus written completely as a macro

It was a delivery system only with no payload

But was an important proof-of-concept

Users often open documents directly from email

# 2000: I Love You

---

Visual Basic Script virus

Appears as email attachment:

- LOVE-LETTER-FOR-YOU.txt.vbs
- The .vbs often hidden on Windows

When executed:

- Damaged many office files
- Sent email out to email address book ***automatically***

Spread worldwide in hours

# 2004: MyDoom

---

Fastest spreading mass mailer virus at the time

- Slows overall internet performance by about 10%
- Slows average web page load times by about 50% percent
- Responsible for approximately one in ten e-mail messages.

Appears as a delivery error, mail error, etc

Includes an attachment that, if clicked on, mails out copies

Also attempted to spread via P2P file sharing Kazaa

Opened a back door for remote control

Attempted to launch a DDOS against the SCO Group's website

# 2005: Sony Rootkit

---

Sony CD's from the 2004-2005 era installed a "Rootkit"

- Rootkit, as name implies, usually installs with elevated access
- Using this elevated access, it can change the OS
- This bypasses usual security such as antivirus, etc
- Also usually very good at being undetectable

Installed at root with an EULA ***that did not mention the software***

In 2005, ***US-CERT ISSUED AN ADVISORY!!!***

Texas, ***under Greg Abbot***, was the first state to sue

# Why is the Sony Rootkit So Bad?

---

In addition to violations of privacy, etc, caused:

- Slowing the system, consuming resources
- False alarms from antivirus

## ***OPENED HOLES FOR ADDITIONAL MALWARE***

- “Stinx-E trojan”

# 2013: Cryptolocker

---

Modern Ransomware

(1980's had a ransomware called CyberAIDS)

Locks up system and uses public key crypto

In addition to fiat currency, accepted BitCoin



# 2016: Mirai

---

Worm that finds vulnerabilities in IoT devices

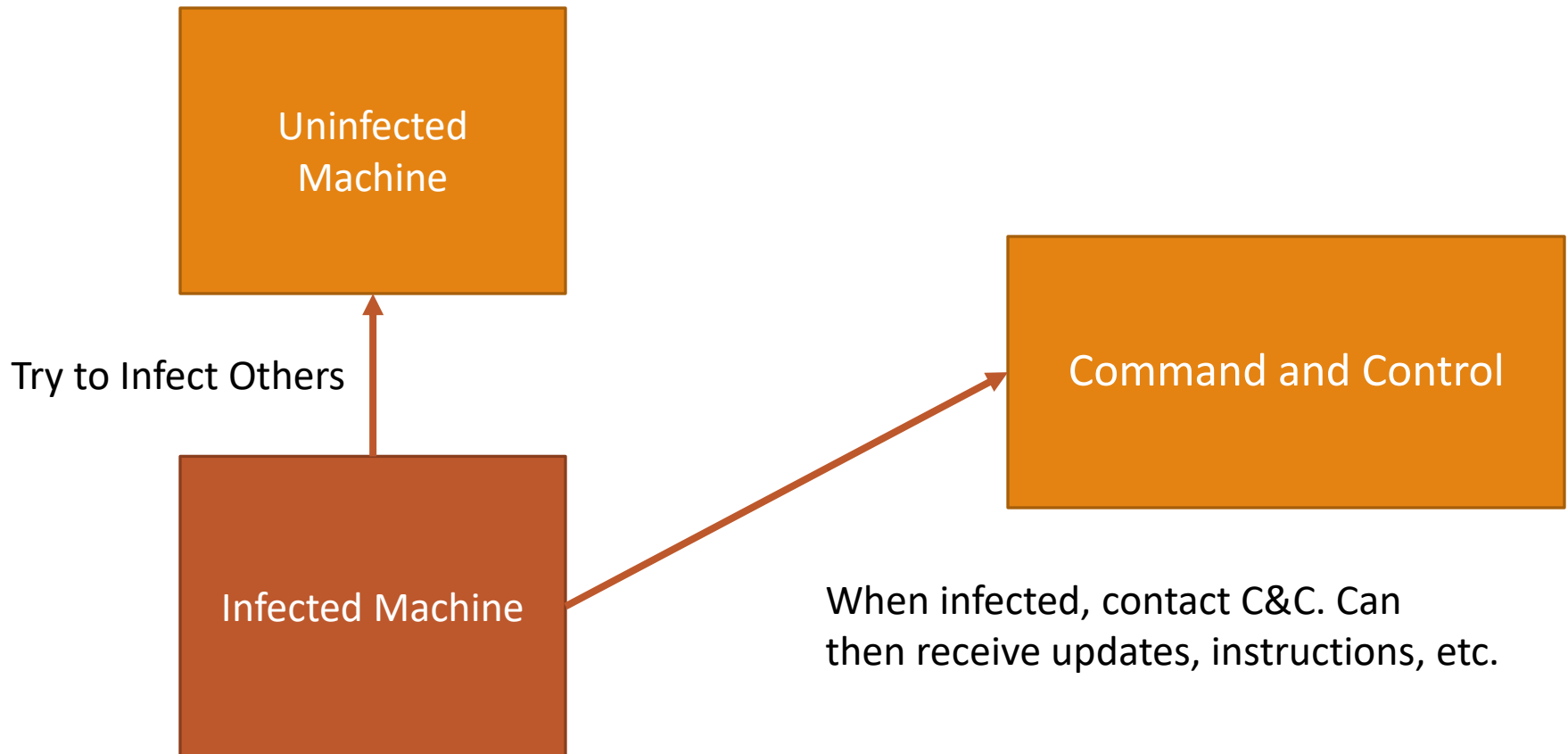
Takes over the device (“Zombie”)

Coordinates all devices with a Command and Control

Launched a powerful DDOS against “krebsonsecurity”

# Command and Control Concept

---



# Malware Defense Strategies

Identify  
and  
neutralize

Identify and neutralize malware before the attack

Mitigate

Mitigate malicious activity during the attack

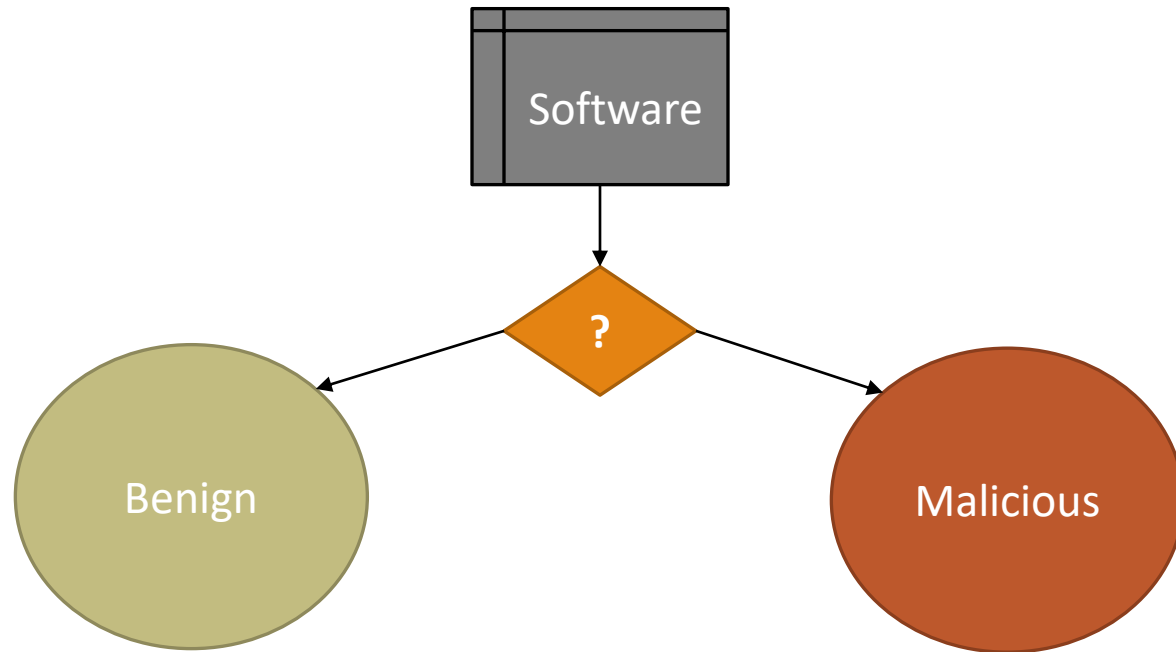
Recover  
or restore

Recover or restore damaged systems after the attack

# Identifying Malware

---

Primarily a ***classification*** problem



# Classification Approaches

---

## Static Analysis

- Analyze the software to categorize it
- Compare against known patterns (signatures)
- Or ~~determine~~ guess how it will behave (heuristics)

## Dynamic Analysis

- Analyze the software's **execution**
- Identify behavior that violates a security policy
- Or, ~~determine~~ guess if behavior is dangerous
- Typically in a “safe” container (emulation or sandboxing)

# Early “Anti Virus”

---

“Virus Bulletin” started in 1989

Still available at [www.virusbulletin.com](http://www.virusbulletin.com)

Used to print **BYTE SEQUENCES** of known viruses

**8 Tunes** - CER: The virus probably originates in Germany and infects COM and EXE files. The length of the virus code is 1971 bytes. When triggered, it will play one out of eight different tunes. The virus attempts to deactivate two anti-virus programs: Bombsquad and Flushot+.

8 Tunes

33F6 B9DA 03F3 A550 BB23 0353 CB8E D0BC ; Offset variable

Virus Bulletin, January 1991

# Virus Advancements

---

Antivirus scanners emerged with “libraries” of virus signatures

In response, viruses became “polymorphic”

- Each infection encrypts virus under a different key
- Decryption engine decrypts virus for operations
- Encryption means that each infection has unique bytes

# Polymorphic Virus Diagram

may soon make this approach unattractive.

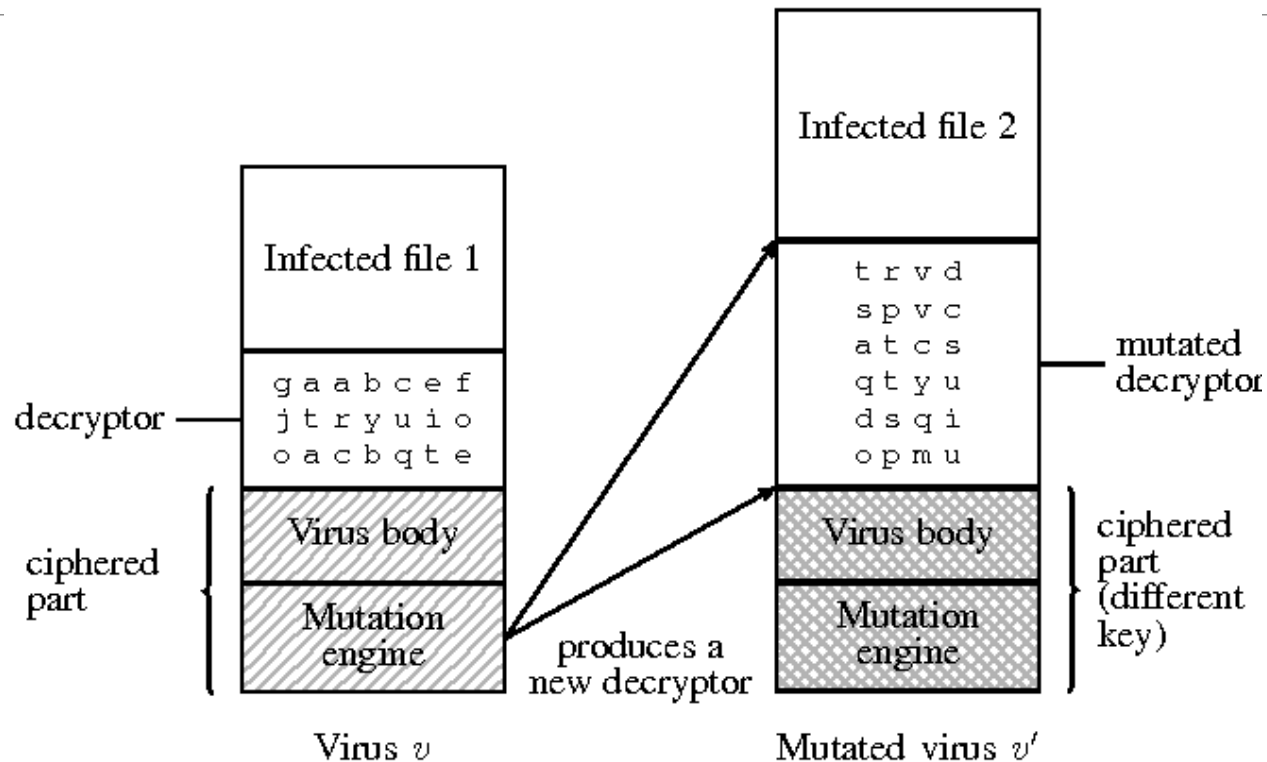


Figure 1. Polymorphic virus infection process

**“Automated extraction of polymorphic virus signatures using abstract interpretation” by Chaumette and Tabary**



# Anti-virus Arms Race

---

## Advanced Signatures

- Signature is not just a byte sequence
- Each “signature” is a mini-program of detection instructions

## Partial Interpreter

- Virus usually takes control early
- Interpret the first bytes to see if its decrypting
- Decrypt and then scan

(Developed in the 1990's)

# Beyond Scanning

---

## Behavior Blockers

- Tries to block bad behavior
- But what counts as “bad”?

## Integrity Checkers

- Checksum files
- Detect unauthorized changes
- But what is authorized?

## Heuristics

- Look for “telltale” signs
- Minimally effective; too many false positives

# Emulation and Sandboxing

---

Emulation ***simulates*** execution

- Default arguments
- Stubbed I/O
- Simulate the “beginning” when viruses activate

Sandboxing runs the software in a virtual environment

- Default arguments
- Usually for a limited amount of time (e.g., 1 minute)
- Observe changes to the filesystem

Problems

- Non-contextual execution including arguments
- Malware that detects sandboxes

# Malware Scanning

---

Antivirus scanning is representative of all malware scanning

Always “behind” the enemy

Signatures can only catch “known” malware

Guesses always have FP and FN

Dynamic execution can be detected/evaded

# Enemy: Halting Problem

---

**Decidability** is a classic computer science problem

Halting Problem:

- Given: a program P and input I
- Can you write a program D that determines if P **halts** on input I
- (Halts, meaning e.g., not stuck in an infinite loop)
- Over the **set of all possible programs**, the answer is **NO**
- (Maybe able to determine for some, but not for all)

Alan Turing proved this in 1936!

***Halting problem proven to extend to any non-trivial characteristic***

# In Other Words

---

There is no program that can detect all malware

Is this just theoretical?

- What if we can detect 99.9999999%?
- What if we can detect all the “important” threats?

# Thoughts from 1995

---

In 1995, Gryaznov wrote, “Scanners of the year 2000”

He discussed Heuristics

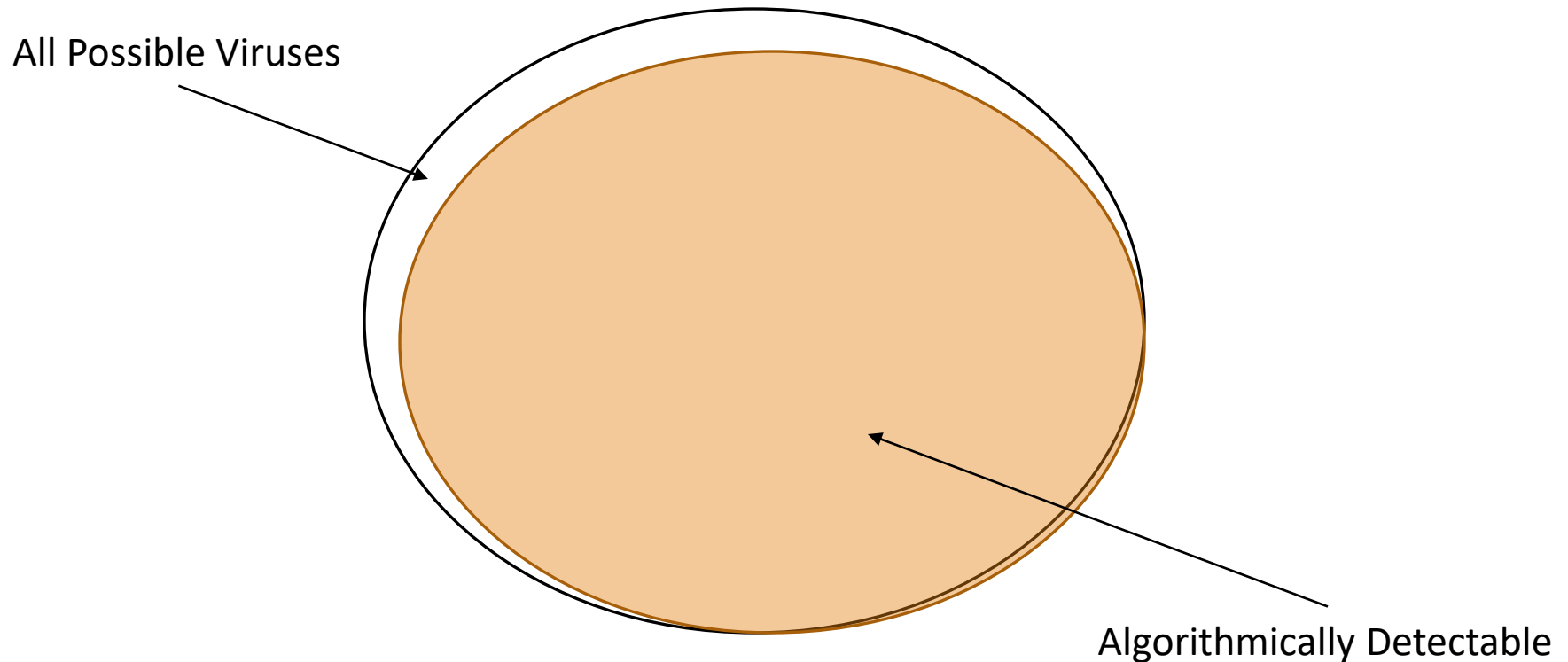
Specifically mentioned the halting problem, but said:

*Fortunately, this does not rule out a possibility of 90 or even 99 per cent reliability. And with the remaining one per cent cases we hopefully shall be able to deal with using our traditional virus signatures scanning technique.*

**GRYAZNOV WAS WRONG.**

# Gryaznov's View

---





# Why It Doesn't Work

---

Gryaznov treats viruses as if they are created at random

Viruses are created by *human beings*

If an antivirus writer creates an algorithm, the adversary adjusts

The adversary moves into the space not detected by the algorithm

# Correct View

---

