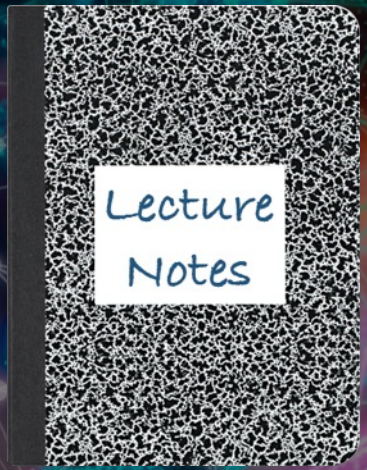



CS 419 – Computer Security

# Week 9: Malware

Paul Krzyzanowski

© 2026 Paul Krzyzanowski. No part of this content may be reproduced or reposted in whole or in part in any manner without the permission of the copyright owner.

# Overview

- **What malware is**
  - Definition, how it spreads, what it does, and how it hides
- **The malware lifecycle**
  - Infection, delivery, persistence, trigger, communication, payload, and propagation
- **Infection and delivery**
  - How malware gets in, and the droppers, loaders, and packers that deliver the payload
- **Persistence through detection**
  - Surviving a reboot, when the payload acts, command and control, what malware does, how it spreads, and how it is caught

# ClickFix Attack: A Fake "I'm Not a Robot" Checkbox

A compromised web page shows a fake "I'm not a robot" checkbox

↓

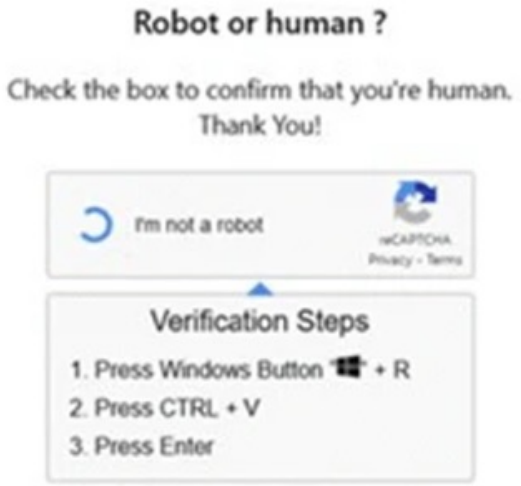
A user's click copies a command. The page says to run it from the Run dialog

↓

The command starts a chain of scripts with the visitor's permissions

↓

**Lumma Stealer** collects browser passwords, session cookies, and wallet data




## Why it worked

The browser blocked the page from running the command directly.

The visitor was persuaded to run it, with the visitor's own permissions.

# What Malware Is

# Malware Is Defined by Intent, Not Operations

**Malware** (malicious software):
Software deliberately used to violate the security interests or authorized policies of the people responsible for a system.

A backup program

Ransomware

Both read and rewrite files

A remote-support tool

Malicious remote control

Both accept instructions from another machine

**Authorization**
comes from the people responsible for the system, not the user at the keyboard

**Intent**
separates a bug that deletes files from code written to delete them

**Privileges**
need not be administrative: a user's own files and browser data are already writable

# Labels Describe Different Properties

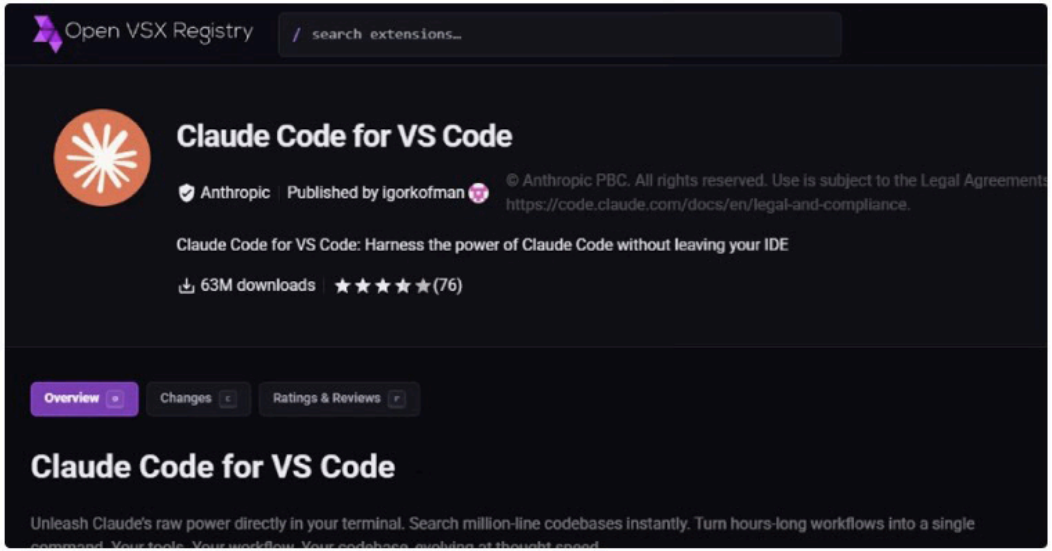
| How it spreads | What it does | How it deceives |
|---|---|---|
| **Virus** inserts its code into a host | **Ransomware** locks data, demands payment | **Trojan** has a legitimate and a covert purpose |
| **Worm** spreads between systems by itself | **Wiper** destroys data beyond recovery | **Backdoor** is an unauthorized way in |
| | **Spyware** covertly collects information | **Rootkit** hides activity from the system |
| | **Infostealer** takes stored credentials | |

**One program can have several labels:**
**a trojanized utility can install an infostealer and a backdoor.**
**A rootkit can hide them.**

The Hacker News

# Official Claude Code Extension on Cursor Marketplace Trojanized with Persistent SSH Backdoor

Ravie Lakshmanan  Oct 30, 2026




Cybersecurity researchers have disclosed that a trojanized version of Anthropic's official Claude Code extension was distributed through the Cursor Marketplace as a legitimate update, attempting to install a persistent SSH backdoor on affected systems.

https://thehackernews.com/2026/10/official-claude-code-vs-code-extension.html

# AI plug-in for ComfyUI

- Robert McMillan, a mid-level manager at Disney
- Downloaded software from GitHub – a plug-in that helped make an AI tool called ComfyUI easier to use
- It worked BUT it was a **trojan**
  - It also had a malicious part
  - Gave access to his 1Password cache and other information on his PC
  - Hackers stole his identity,  logged into his work Slack account & downloaded data
  - Dumped all of Robert's logins & passwords online
  - Robert was fired

# Backdoors

**Backdoor**:

- Hidden mechanism to bypass normal authentication or access controls
- Backdoors in malware can provide future access to the attacked system

- Ken Thompson's modified C compiler installed a back door to `login`
- A modification to the XZ Utils compression library discovered in 2024 enabled remote command execution

**Backdoors may be built in or added later via an exploit**

# Legitimate Backdoors

- Backdoors may be installed for legitimate purposes, such as maintenance
  - The author of *sendmail* installed a backdoor to fix bad configurations
- But attackers can discover and exploit these backdoors
  - The Morris attack checked for the *sendmail* backdoor
  - A 2024 cyberattack on broadband providers (AT&T, Verizon, …) provided access to information from systems the federal government uses for court-authorized network wiretapping requests

ars TECHNICA

BROADBAND PROVIDERS HACKED

**Reports: China hacked Verizon and AT&T, may have accessed US wiretap systems**

Chinese hackers were in networks of major ISPs "for months," WSJ reports.

JON BROOKIN – OCT 7, 2024 5:45 PM · 118

Credit: Getty Images | Bill Hinton Photography

Chinese government hackers penetrated the networks of several large US-based Internet service providers and may have gained access to systems used for court-authorized wiretaps of communications networks, The Wall Street Journal reported Saturday. "People familiar with the matter" told the WSJ that hackers breached the networks of companies including Verizon, AT&T, and Lumen (also known as CenturyLink).

"A cyberattack tied to the Chinese government penetrated the networks of a swath of US broadband providers, potentially accessing information from systems the federal government uses for court-authorized network wiretapping requests," the WSJ wrote. "For months or longer, the hackers might have held access to network infrastructure used to cooperate with lawful US requests for communications data, according to people familiar with the matter."

These "attackers also had access to other tranches of more generic Internet traffic," according to the WSJ's sources. The attack is being attributed to a Chinese hacking group called Salt Typhoon.

# Millions of PC Motherboards Were Sold With a Firmware Backdoor

WIRED

**Hidden code in hundreds of models of Gigabyte motherboards invisibly and insecurely downloads programs—a feature ripe for abuse, researchers say.**

Andy Greenberg • May 31, 2023

Hiding malicious programs in a computer's UEFI firmware, the deep-seated code that tells a PC how to load its operating system, has become an insidious trick in the toolkit of stealthy hackers. But when a motherboard manufacturer installs its own hidden backdoor in the firmware of millions of computers—and doesn't even put a proper lock on that hidden back entrance—they're practically doing hackers' work for them.

Researchers at firmware-focused cybersecurity company Eclypsium revealed today that they've discovered a hidden mechanism in the firmware of motherboards sold by the Taiwanese manufacturer Gigabyte, whose components are commonly used in gaming PCs and other high-performance computers. Whenever a computer with the affected Gigabyte motherboard restarts, Eclypsium found, code within the motherboard's firmware invisibly initiates an updater program that runs on the computer and in turn downloads and executes another piece of software.

…

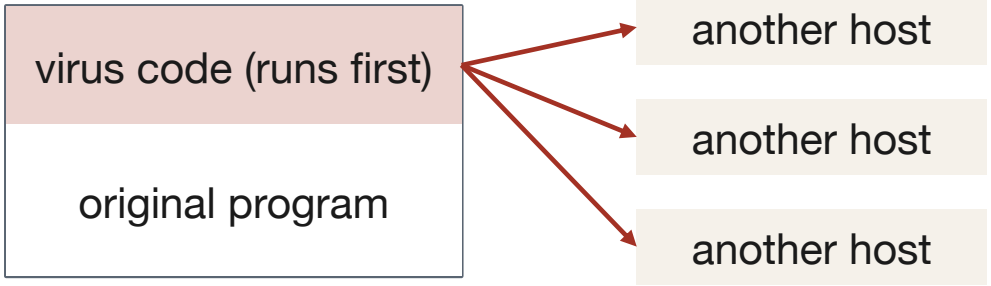
"If you have one of these machines, you have to worry about the fact that it's basically grabbing something from the internet and running it without you being involved, and hasn't done any of this securely," says John Loucaides, who leads strategy and research at Eclypsium.

https://www.wired.com/story/gigabyte-motherboard-firmware-backdoor/

# Viruses Replicate Through a Host

A **virus** replicates by inserting its code into a host, such as an executable, document with macros, or boot sector on a disk. The host runs the virus when it runs.

**An infected host program**




*The virus runs whenever the host runs*

**The host is part of the definition**

A credential thief that does not insert itself into another program or file is not a virus

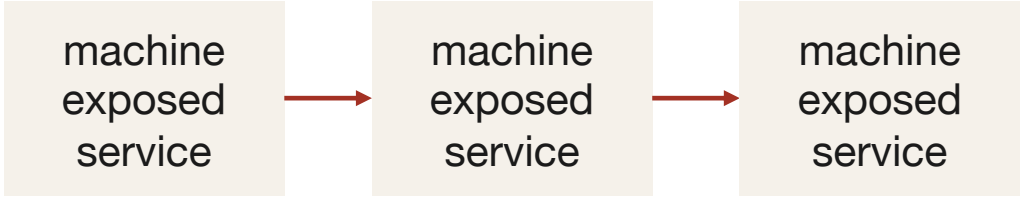
**Origin of the term: November 1983**

Fred Cohen inserted code into a Unix utility.
Everyone who ran it ran his code with their own permissions.

# Worms Spread Between Systems on Their Own

A **worm** is a self-replicating program that spreads between systems without inserting its code into a host program or document.

### Each infected machine looks for another




*Exploit the service, copy over, repeat*

### Worm versus virus

The host is the defining difference. Spreading without user action is common, not an absolute rule.

### Propagation and persistence are independent

A worm can spread from memory and vanish at reboot, while a backdoor can persist for years

"All the News That's Fit to Print"

# The New York Times

**Late Edition**
New York: Today, windy, occasional rain. High 58-64. Tonight, showery and mild. Low 52-55. Tomorrow, showers, breaking clouds. High 58-62. Yesterday: High 65, low 45. Details are on page 47.

VOL.CXXXVIII . . . No. 47,680 Copyright © 1988 The New York Times NEW YORK, SATURDAY, NOVEMBER 5, 1988

50 cents beyond 75 miles from New York City, except on Long Island. 35 CENTS

## *Author of Computer 'Virus' Is Son Of N.S.A. Expert on Data Security*

### *Cornell Graduate Student Described as 'Brilliant'*

By JOHN MARKOFF

The "virus" program that has plagued many of the nation's computer networks since Wednesday night was created by a computer science student who is the son of one of the Government's most respected computer security experts.

The program writer, Robert T. Morris Jr., a 23-year-old graduate student at Cornell University whom friends describe as "brilliant," devised the set of computer instructions as an experiment, three sources with detailed knowledge of the case have told The New York Times.

The program was intended to live innocently and undetected in the Arpanet, the Department of Defense computer network in which it was first introduced, and secretly and slowly make copies that would move from computer to computer. But a design error caused it instead to replicate madly out of control, ultimately jamming more than 6,000 computers nationwide in this country's most serious computer "virus" attack.

The dent's program jammed the computers of corporate research centers including the Rand Corporation and SRI International, universities like the University of California at Berkeley and the Massachusetts Institute of Technology as well as military research centers and bases all over the United States.

**Meeting with the Authorities**

The virus's creator could not be reached for comment yesterday. The sources said the student flew to Washington yesterday and is planning to hire a lawyer and meet with officials of the Defense Communications Agency, in charge of the Arpanet network.

Friends of the student said he did not intend to cause damage. They said he created the virus as an intellectual challenge to explore the security of computer systems.

His father, Robert T. Morris Sr., has written widely on the security of the Unix operating system, the computer master program that was the target of the son's virus program. He is now chief scientist at the National Computer Security Center in Bethesda, Md., the arm of the National Security Agency devoted to protecting computers against outside attack. He is most widely known for writing a program to

## 'VIRUS' ELIMINATED, DEFENSE AIDES SAY

### Crucial Computer Networks Said to Be Impenetrable

By MICHAEL WINES
Special to The New York Times

WASHINGTON, Nov. 4 — Defense Department officials said today that they had eliminated an electronic "virus" that played havoc with an un-

## POLAND IS BUYING 3 BOEING AIRLINERS FOR $220 MILLION

### EAST BLOC ORDER A FIRST

### Sale to Be Financed Through a Lease-Purchase Accord With Western Banks

By AGIS SALPUKAS

The Boeing Company received an order yesterday from the national airline of Poland, the first order for advanced American aircraft from an Eastern bloc country.

The order from the LOT airline is for three 767 wide-bodied aircraft and is worth about $220 million. The transaction is to be financed through a lease-purchase agreement with Western banks, under which the airline will own the planes after 12 years.

Airline officials, at a news conference at the Polish Consulate in New York yesterday, would not identify the Western banks involved in the transaction.

The airline is state-owned and Poland's troubled economy is deeply in debt. But the new planes will bring the carrier significant savings on fuel, and the modern, more spacious aircraft could attract more bookings from Western travelers.

**Planes Can Be Repossessed**

The banks are apparently relying on those factors for assurance that the airline can make its lease payments.

# *MOSCOW SUSPENDS PULLOUT OF ITS AFGHANISTAN FORCES; CHARGES VIOLATIONS OF PACT*

### *U.S. Expresses Disappointment*

President Reagan said yesterday that he was disappointed by the Soviet Union's decision to suspend the withdrawal from Afghanistan. The State Department said the suspension was disturbing.

Marlin Fitzwater, the White House spokesman, said the Soviets' actions "can only increase tensions in the region and raise speculation that they aren't going to live up to the Geneva accords."

But Administration officials nevertheless drew attention to Moscow's statement that the Soviet Union still intends to adhere to the accords, which call for the troop withdrawal to be complete by Feb. 15.

*Article, page 4.*

Associated Press

Aleksandr A. Bessmertnykh, a Soviet Deputy Foreign Minister, announced suspension of troop withdrawal from Afghanistan.

## BETTER ARMS SENT

### Soviets Hint at a Delay Past Feb. 15 Deadline for Full Withdrawal

By PHILIP TAUBMAN
Special to The New York Times

MOSCOW, Nov. 4 — The Soviet Union said today that it was suspending the withdrawal of its troops from Afghanistan and was supplying the Afghan Army with more powerful weapons because of stepped-up military activity by guerrilla forces.

Moscow left open the option of delaying its withdrawal beyond a February deadline for completing the removal of Soviet troops.

Aleksandr A. Bessmertnykh, a Deputy Foreign Minister, said the withdrawal — which started on May 15, paused on Aug. 15 and had been expected to resume later this month — was being delayed because of a worsening military situation in Afghanistan.

**Vows to Carry Out Accords**

He said at a news conference that "the Soviet Union intends to carry out

## *Unemployment Declines to 5.2%, Matching Lowest Rate Since '74*

By ROBERT D. HERSHEY Jr.

# Story: The Morris Worm

**November 2, 1988.** Robert Tappan Morris, Jr., a Cornell graduate student, released a self-propagating program from MIT. It was the first major attack on the Internet.

### Guess passwords

List of common passwords, account names, `/usr/dict/words`

### Trusted hosts

`.rhosts` and `hosts.equiv` gave *rsh* access without a password

### fingerd overflow

Code injection of a 99-line C program via `gets()` that fetched the rest

### sendmail DEBUG

A debugging mode that ran a command

### Why it brought machines down

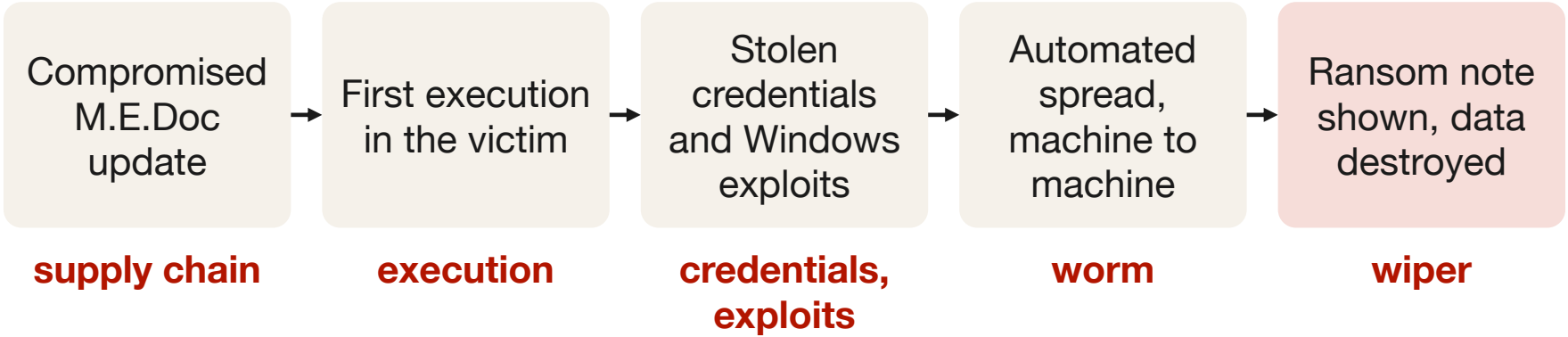
A check for existing copies was skipped 1 out of 7 times, so machines ran many copies and the load made them unusable

### The aftermath

First conviction under the Computer Fraud and Abuse Act, 1990: probation, community service, and a fine. DARPA funded the first CERT.

# NotPetya: Several Labels on One Incident

**June 2017.** Microsoft traced delivery through the Ukrainian M.E.Doc accounting software's update channel




**Several labels on one incident**
Supply chain compromise, credential theft, worm propagation, and a wiper payload

**Each label suggests a defense**
Block the update to prevent initial execution.
Restrict movement to limit the spread.

# Evolution of Malware

## Dominant Threats

- **1980s-1990s: Boot sector viruses, macro viruses**
  - Motivation: Notoriety, pranks, experimentation
  - Examples: Brain, Michelangelo, Melissa
- **2000s: Worms, spam bots, adware**
  - Motivation: Monetization begins
  - Examples: ILOVEYOU, Blaster, Code Red
- **2010s: Ransomware, APTs, banking trojans**
  - Motivation: Serious profit, espionage
  - Examples: CryptoLocker, Stuxnet, Zeus
- **2020s: Supply chain, fileless, polymorphic, AI-enhanced**
  - Motivation: Sophisticated, targeted, persistent
  - Examples: SolarWinds, DarkSide, LockBit 3.0

# The Malware Lifecycle

| | |
|---|---|
| **Infection** | How the malware arrives and first executes |
| **Delivery** | A dropper or loader deploys the actual payload, often in stages |
| **Persistence** | Arranging to survive logouts and reboots |
| **Trigger** | The condition under which the payload activates |
| **Communication** | Contacting an operator for instructions and returning results |
| **Payload** | What the malware does: steal, encrypt, control, or sabotage |
| **Propagation** | Spreading to other systems, for malware built to do so |

Malware can omit, repeat, or combine these activities

# Infection: How Malware Gets In

# Three Ways Malware Gets In

**1. Exploit a flaw**

Network service, browser, document viewer, or media handler

**2. Physical access**

Removable drive, maintenance laptop, or file carried across an air gap

**3. Persuade a user**

Install software, run a command, open an attachment, or click a link

Not exclusive, and not the only ways in: stolen administrator credentials and compromised update services deliver code through approved channels

**Initial access**

The first foothold in the target

**Execution**

Running attacker-chosen instructions

**Privilege escalation**

Greater authority on the machine

**Lateral movement**

Use one machine to reach others

# Exploiting a Running Program

**Input**: file, message, or network data → **A processing flaw**: memory corruption or command injection → Code runs with the process's privileges → The payload installs itself

## An exploit

Uses a vulnerability to make a system perform an unintended operation. It gives the attacker a way to run code.

## Process privileges only

Exploiting a document renderer does not give control of the kernel.

Escaping a sandbox needs a second flaw

**Repairing the vulnerability closes that entry path.**
**It does not remove a payload that has already installed itself.**

# Zero-Day Vulnerabilities & Exploits

## A zero-day vulnerability

Exploited before a protective patch is available.

A **zero-day attack** uses such a flaw.

**Zero day** = developers and system administrators have zero days to come up with a fix once the vulnerability is made known.

System administrators cannot take preventive measures to guard against them.
Software developers don't know about them and have not developed patches.

---

## N-Day Exploit

N = # days since the vulnerability has been public

Patch is posted → Attackers examine the patch to find the original bug → Craft an exploit against the patched vulnerability → Attack unpatched systems

# Code Red and Slammer: Automated Propagation

### Code Red, July 19, 2001

Buffer overflow in Microsoft's IIS web server

More than **359,000** machines infected in **14 hours**

Every vulnerable server was reachable through the interface it offered to web clients

### SQL Slammer, January 25, 2003

Buffer overflow in SQL Server; the whole worm fit in one **376-byte** UDP packet

More than **90 percent** of vulnerable hosts infected in **10 minutes**

A patch had been available before the outbreak

**Each infected host becomes another scanner.**
**A defense that starts after the first infection competes with automated replication.**

# The Human Vulnerability

## Why attack technology when you can attack people?

**Social engineering**: manipulate people into doing something that is against their best interest (e.g., breaking security procedures)

- Why it's effective
  - Humans are trusting by nature; we want to be helpful
  - We're busy and make quick decisions
  - We're scared, rushed, or excited

"It's easier to trick someone into giving you their password than to hack the system" – Kevin Mitnick (approximate quote)

# Social Engineering, Phishing, and Pretexting

**Phishing**

a deceptive message or site that impersonates a trusted source

**Spear phishing**

targets a particular person with details chosen to make the request plausible

**Pretexting**

an invented situation that makes the requested action look necessary

**The Lumma page's pretext: proving you're a human requires running this command**

**AI automates the research and writing that once limited spear phishing**

# Psychological Principles Attackers Exploit

**Authority**
- "This is IT support…"
- "The CEO needs this wire transfer now"

**Urgency**
- "Your account will be closed in 24 hours"
- "Your license will be suspended"

**Trust**
- "Your colleague sent you this rocument"
- Impersonating trusted brands

**Fear**
- "Your computer is infected"
- "Legal action will be taken…"

**Curiousity**
- "Photo from last night"
- "From a secret admirer"

**Greed**
- "You've won a prize"
- "Free gift card"

# Trust Signals and Their Limits

### What a convincing message has

- A logo
- A deadline
- Personal details
- Forged sender
- A real conversation history (compromised account)
- HTTPS (encrypts only the connection)

### Mismatches that deserve scrutiny

- A verification page says to run a command in a system utility
- A document says its contents require disabling a security control
- A payment or password change is directed outside the usual process
- A familiar sender asks for secrecy or urgency
- The displayed link or sender differs from the real one

**Verify through a channel the message did not supply**

# The Booking.com Campaign

Reservation-related email to hotel staff, impersonating Booking.com

↓

Link to a fake verification page

↓

ClickFix: have the user copy & run a command to "authenticate"

↓

Lumma and other credential thieves

**December 2024 to March 2025**

Reported by Microsoft on March 13, 2025

**Correct facts, wrong sender**

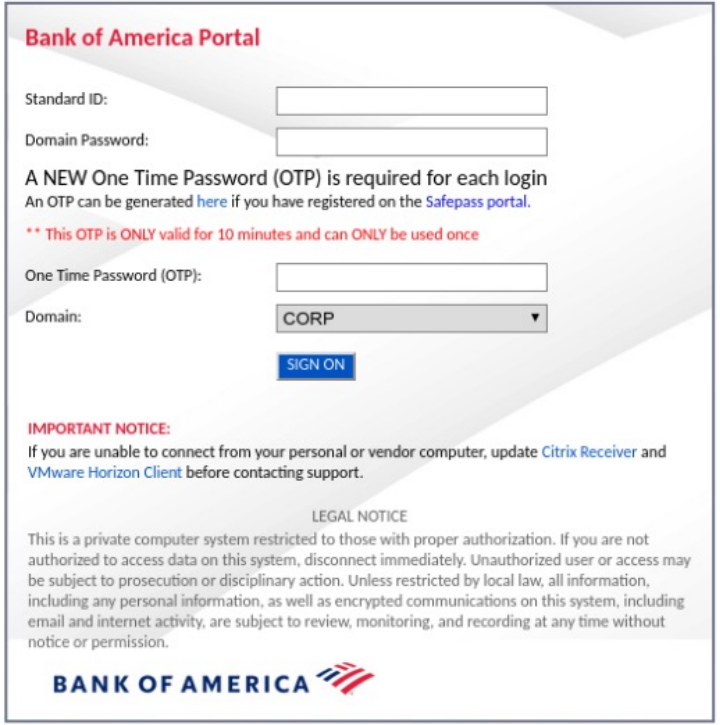
Reservation data makes the next fraudulent message credible

**The safeguard**

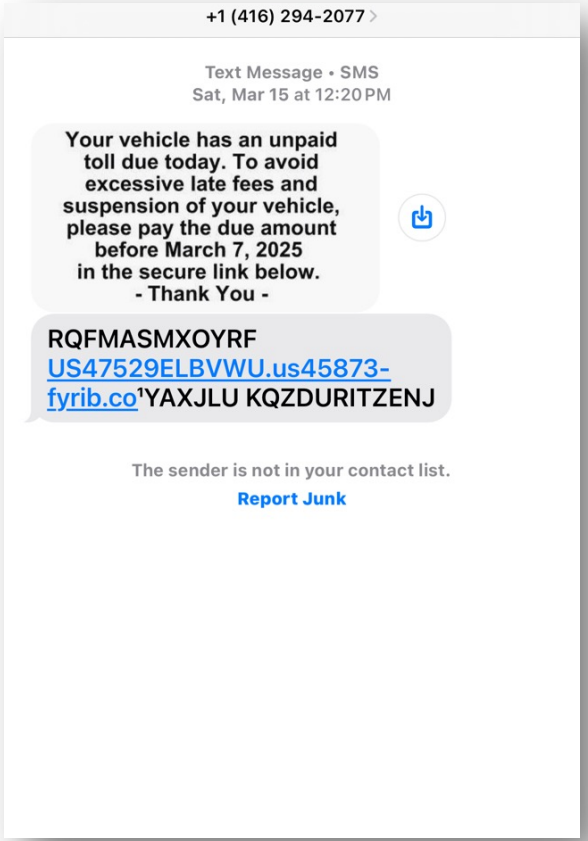
Check the booking directly, not via the link

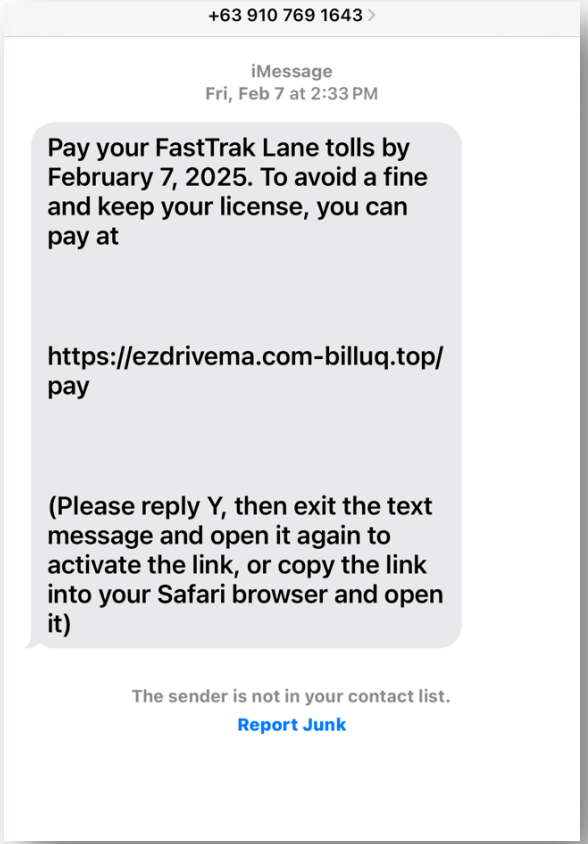
# Phishing: Voice & SMS too

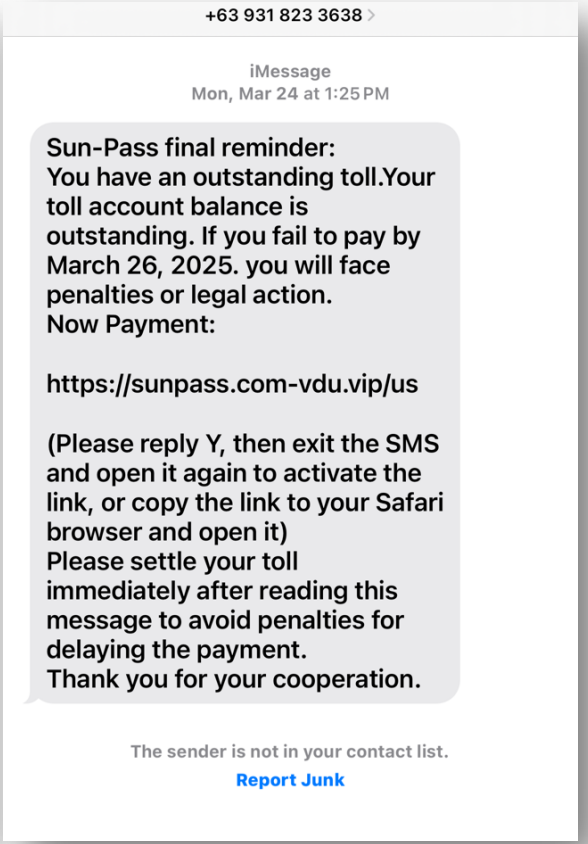
- Vishing (voice phishing)
  - Microsoft tech support
  - IRS refund
  - Bank withdrawal confirmation
- Smishing (SMS phishing)
  - People trust text more than email
  - Mobile devices have smaller screens – harder to verify the sender
  - Sense of urgency



# Toll Payment

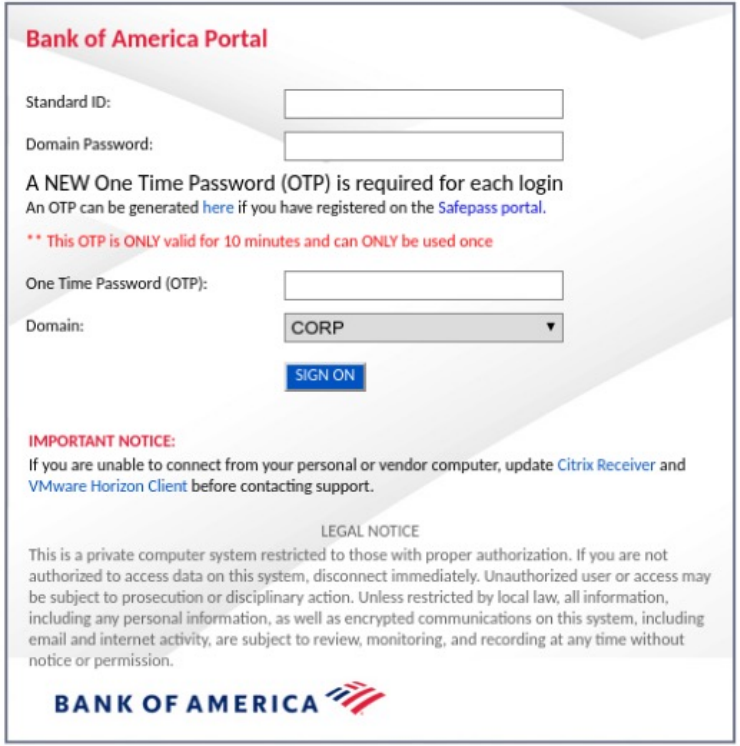


# Voice phishing (vishing)

- 2020 saw a lot of email attacks to trick work-at-home employees to divulge access credentials to their corporate network
- Hackers-for-hire offer voice phishing services
  - Created lots of company-branded phishing pages targeting some of the world's biggest companies
  - Place calls to employees working at home
  - Explain that they are calling from the IT department to troubleshoot VPN issues
  - Goal: convince employee to divulge credentials
  - Hackers may create corporate LinkedIn profiles for deception



# Typosquatting & Similar Domains

- Exploit mistakes users might make while typing
- Typosquatting: Registering domains similar to legitimate sites

**Typosquatting**
- gooogle.com (extra 'o')
- amazom.com (n→m)

**Character substitution**
- g00gle.com (o→0)
- paypa1.com (L→1)

**Omitted characters**
- gogle.com (missing 'o')
- amazn.com (missing 'o')

**Wrong domain**
- google.net (.com→.net)
- microsoft.org

**Combosquatting**
- microsoft-support.com
- amazon-security.com

SC Media
A CRA Resource

DevOps, Malware, DevSecOps

# Typosquatting campaign targets financial sector Linux, macOS systems

March 5, 2025

By Steve Zurier

(Adobe Stock)

The Register®

# Ongoing typosquatting campaign impersonates hundreds of popular npm packages

## Puppeteer or Pupeter? One of them will snoop around on your machine and steal your credentials

Jessica Lyons

Tue 5 Nov 2024 // 16:28 UTC

An ongoing typosquatting campaign is targeting developers via hundreds of popular JavaScript libraries, whose weekly downloads number in the tens of millions, to infect systems with info-stealing and snooping malware.

The npm supply chain attack appears to have originated in October, and we've seen three different security shops sound the alarm on this novel typosquatting effort that uses Ethereum smart contracts for command-and-control (C2) operations.

In this case, typosquatting involves a criminal publishing malicious npm packages with names that look like legitimate ones, but are just slightly off by a letter or two –

# PyPI Halts Sign-Ups Amid Surge of Malicious Package Uploads Targeting Developers

The Hacker News

March 29, 2024

The maintainers of the Python Package Index (PyPI) repository briefly suspended new user sign-ups following an influx of malicious projects uploaded as part of a typosquatting campaign.

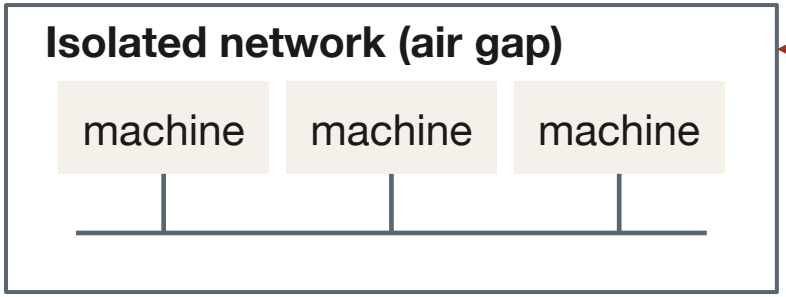
PyPI said "new project creation and new user registration" was temporarily halted to mitigate what it said was a "malware upload campaign." The incident was resolved 10 hours later, on March 28, 2024, at 12:56 p.m. UTC.

Software supply chain security firm Checkmarx said the unidentified threat actors behind flooding the repository targeted developers with typosquatted versions of popular packages.

"This is a multi-stage attack and the malicious payload aimed to steal crypto wallets, sensitive data from browsers (cookies, extensions data, etc.), and various credentials," researchers Yehuda Gelb, Jossef Harush Kadouri, and Tzachi Zornstain said. "In addition, the malicious payload employed a persistence mechanism to survive reboots."

https://thehackernews.com/2024/03/pypi-halts-sign-ups-amid-surge-of.html

# Removable Media and Air Gaps




removable drive

maintenance laptops, project files, and drives cross the air gap

**Air gap**

Separates a network from other networks. Removable storage becomes the entry point.

**Stored bytes do not execute**

Execution needs a mechanism: a user, a startup feature, or a software flaw

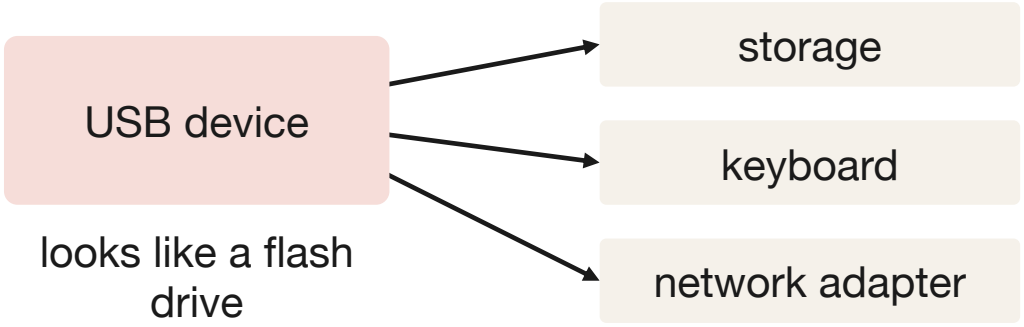
**Stuxnet, discovered in 2010**

Spread via removable drives: a Windows shortcut-file flaw (viewing the folder was enough) and, earlier, AutoRun

# Malicious Flash Drives

- USB Drop Attack: hope a curious person plugs the drive in
- Malicious software & links
- Data leakage: easy to lose the device – others get your data
- Useful for bridging between air-gapped systems
- Malicious firmware
- USB Killer – destroy the USB port via high voltage

# Impersonating USB Devices with BadUSB




At connection, the device reports its own interfaces, and the operating system believes it.

**BadUSB**

Alters or abuses USB-device firmware so the device performs an unexpected function.

**Keystrokes run as the user**

Injected input has only the privileges of the logged-in session

**Disabling AutoRun, blocking executables, and blocking storage do not stop a device that claims to be a keyboard**

# USB Rubber Ducky In Action

1. Plugged into unlocked computer
2. Detected as keyboard
3. Opens command prompt (Win+R)
4. Enters command to download malware
5. Executes malware
6. Clears command history
7. Closes windows

**USB Rubber Ducky**

- $100 from shop.hak5.org
- Typical attack: <10 seconds while user is away from desk

**Why it works**

- Executes faster than human can react
- Looks like a legitimate keyboard to OS
- No "files" to scan (it's a HID device)
- User may not notice a brief screen flicker

# Documents That Execute Code

**Macros**

a program the application runs; it needs macros enabled, not a vulnerability

**Parsing flaws**

malformed content corrupts memory or reaches an unsafe interpreter; disabling macros does not help

**Scripts and links**

PDF viewers run embedded JavaScript, and any document can carry a link or QR code to a phishing page

**March 26, 1999: Melissa**

A Word macro used Outlook to mail the document to the first fifty addresses in each victim's address book

**Office now blocks macros by default in files from the Internet**

# Drive-by Downloads and Zero-click Attacks

**Drive-by download**

Delivers unwanted software through a visited web page, often by exploiting the browser.

**Zero-click attack**

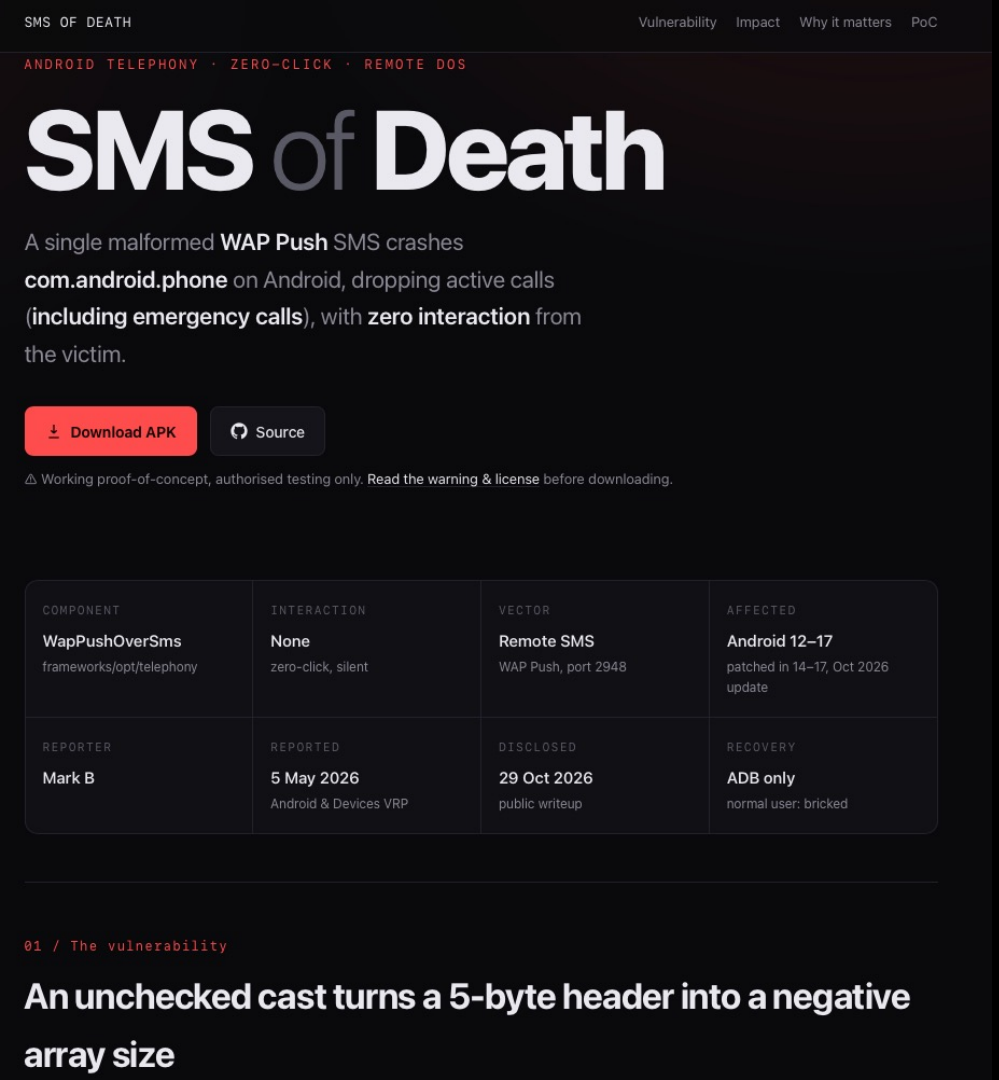
Requires no actions by the user.

**2021: FORCEDENTRY**

Pegasus spyware delivered through iMessage.

A PDF disguised as a GIF reached the JBIG2 image decoder, where an integer overflow led to an out-of-bounds write. ***Automatic processing made it zero-click.***

**Zero-click describes the user interaction required.**
**A zero-click attack can use an old, long-patched flaw on an unpatched device**




When receiving a WAP push message, Android reads the header as variable-length (5-byte) unsigned int and casts it to a signed int.

0x80000000 → -2147483648

This causes an exception.

Nothing appears in messages. No number is displayed. Active calls drop. No service until the service restarts.

The carrier does not get acknowledgements, so it retries, crashing the phone repeatedly.

https://smsofdeath.com

https://github.com/MarkBlah/smsofdeath

# Delivery: Droppers and Loaders

# The Lumma Delivery Chain

| Component | Role in delivery | Trust boundary crossed |
|---|---|---|
| Fake verification page | Persuades the user to start execution | A web page gains influence over an OS action |
| Command run by the user | Starts an interpreter that fetches code | Untrusted text becomes an instruction |
| `mshta`, then PowerShell scripts | Download and prepare later components | A network response becomes executable content |
| Loader | Starts the payload in another process | The visible program name differs from the code running |

**One command from the visitor.**
**The credential thief arrives only at the end of the sequence.**

# Why Delivery Uses Stages

**Multi-stage delivery**
Divides the work across a sequence of components

## Why attackers stage

- Small first stage fits in an attachment
- Later stage matched to the target machine
- Payload replaced without changing the lure

## What staging gives defenders

- Each download can be interrupted
- An unavailable server stops the chain
- Launches and connections are events to inspect

**A delivery stage is not an intrusion stage: the whole Lumma chain accomplished only execution. Legitimate installers stage too.**

# Droppers, Downloaders, Loaders, and Packers

### Dropper

Contains another payload and places it on the target.

Bundling both allows delivery even when the machine cannot reach a server.

### Downloader

Retrieves a later component from the network.

The first file stays small and can be changed independently, but the request can be blocked.

### Loader

Prepares and starts another component, as the OS loader does for ordinary programs.

It can decode a payload in memory without writing a file.

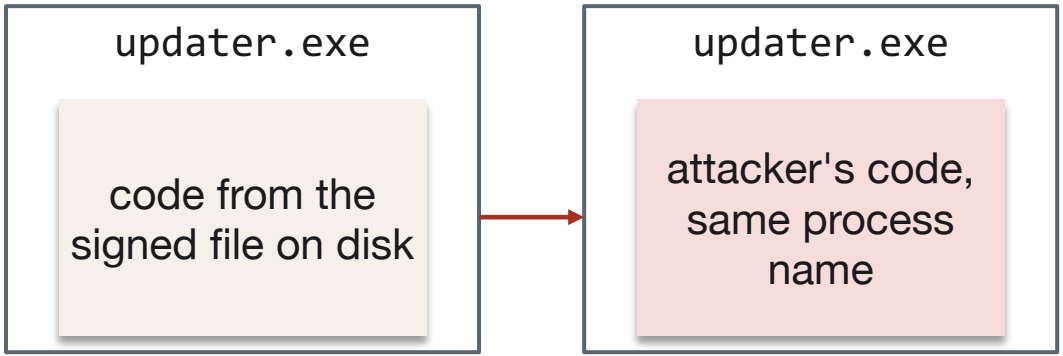
### Packer

transforms an executable through compression or encryption and adds code that restores it at runtime.

A scanner sees only the transformed bytes.

# Process Injection and Process Hollowing

**Process hollowing**




The process list shows a familiar name while different instructions execute

**Process injection** runs code inside another process

**Process hollowing** replaces the code of a process started from a legitimate program.

**A signature covers the file**

not the code placed in the process after it started

**Injection needs access**

A process cannot write into every other process on the machine. Access controls and containment restrict the target.

# Fileless Malware and Living off the Land

## Fileless malware

Execution that does not depend on a conventional payload executable on disk: *memory-resident code, interpreter scripts, instructions stored in configuration data*

## Living off the land

Using programs already on the system, such as PowerShell and remote-management tools, for the attacker's work

## Memory is volatile

Code only in RAM disappears at reboot, so persistence still needs something stored that reloads it, such as a startup entry that fetches a script

## A signed interpreter proves nothing about the script

The same PowerShell can run a scheduled backup or steal a browser profile

**Interpreters still leave traces:**
**cached scripts, command history, logs, and memory paged to disk**

# Persistence

# Persistence Is a Trigger Plus a Registration Point

**Persistence** allows malicious code to resume execution after an interruption such as logout, process termination, or reboot. It combines stored code or configuration with an event that loads it.

**A trigger event**
the system boots, a user logs in, a schedule fires, a shell or application starts, or a library is loaded

**A registration point**
a service, startup entry, scheduled task, profile file, template, or library path

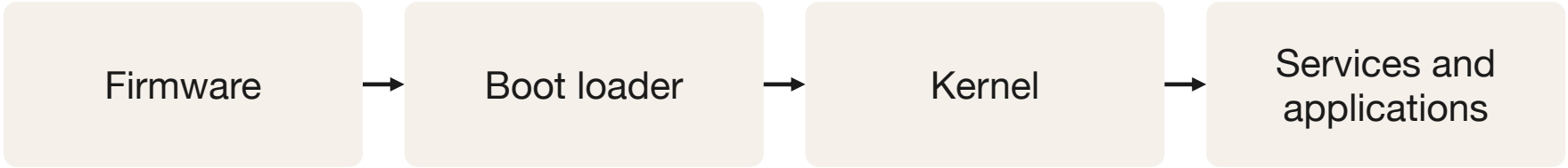
**Not always needed**
A one-run credential thief has nothing to resume, and an unpatched flaw can be exploited again

**Cleaning a machine means removing both the running program and the settings that would start it again**

# Where Code Is Registered to Run Again

| Code runs when... | Windows | Linux and macOS |
|---|---|---|
| The system boots | Windows service | systemd unit, launchd daemon |
| A user logs in | Run key, Startup folder | launchd agent, login item |
| A schedule fires | Scheduled task | cron job, systemd timer |
| The user opens a shell | PowerShell profile | .bashrc, .zshrc, .profile |
| An application starts | Office template, extension | Extension, Office template |
| A library is loaded | DLL search-order hijacking | LD_PRELOAD, path hijacking |

# The Boot Sequence




Each stage loads and starts the next. Code that runs earlier controls what runs later.

**Firmware**
Software in nonvolatile storage that initializes the hardware and starts the boot loader

**Boot loader**
S program that loads the kernel into memory and transfers control to it

**Malware that runs before the kernel runs before the kernel's protections and before security software**

# Boot Sector Viruses and Bootkits

**Boot sector virus**

Replaces the small program a PC executes from a disk at startup then infects other disks or files

**Bootkit**

Modifies or abuses the boot process to gain early execution.

Its files sit on the disk partition used for startup.

**Firmware implant**

Modifies code or data in firmware storage itself. Replacing the disk does not remove it.

**January 1986: Brain – the first PC virus (DOS)**

While in memory it returned the original boot sector to any inspection utility, so the inspector ran normally but received false data.

**Early execution lets a bootkit modify later code in memory, including code with a valid signature on disk**

# Secure Boot Verifies Signatures

Firmware checks the boot loader's signature → Boot loader checks the kernel's signature → Kernel checks each driver's signature

Trusted public keys and a revocation list live in the firmware

**Secure Boot**

Verifies each boot component against an authorization policy before letting it execute

**A signature is not safety**

Old signed boot code with a known flaw still passes unless the policy has revoked it

**Fixing a vulnerable boot loader also requires revoking the old one and updating recovery media that still has it**

# Story: BlackLotus

Installer runs with administrator rights

↓

Places files on the EFI System Partition, the disk partition used for UEFI startup

↓

Supplies older signed boot components with a flaw fixed in January 2022

↓

Secure Boot accepts them: their signatures are valid and they were never revoked

↓

The bootkit disables Windows checks on executable code and installs later components

**March 2023**

ESET, a Slovak security company, analyzed **BlackLotus**, a bootkit sold on forums since October 2022

**The lesson**

Secure Boot did what it was designed to do. The policy had not caught up with the vulnerability.

# Code-execution flaw in UEFI firmware affects most servers on planet Earth

ars technica

## Critical flaw in EDK2 allows attackers to execute code early in the boot process.

Dan Goodin• October 27, 2026

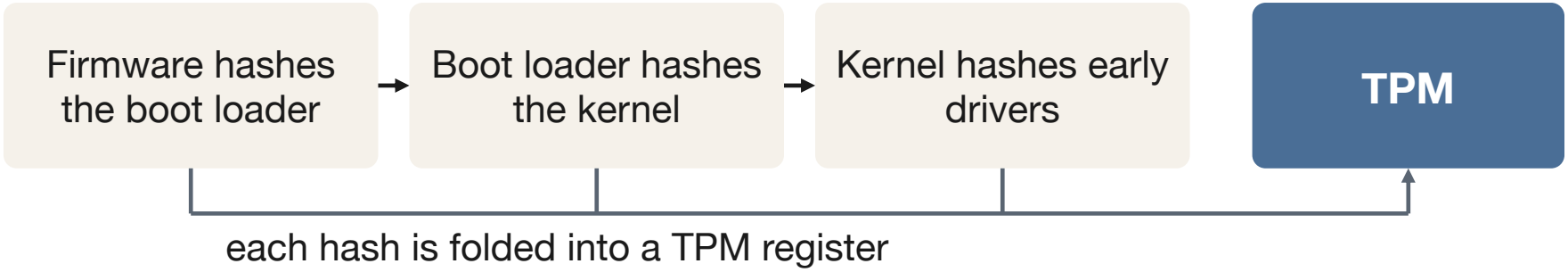
Defenders across the world face the burden of responding to yet another botched code-execution vulnerability, this time in boot-time firmware used by most of the world's server vendors and data centers.

The vulnerability, with a severity rating of 9.5 out of 10, resides in the TianoCore EDK II, an open source codebase that hardware makers and cloud providers use to create the UEFI firmware that runs during the boot-up sequence. Last year, a representative of the UEFI Forum called EDK2 the "preeminent solution / implementation of the UEFI specifications." Virtually all UEFI suppliers (sometimes still called IBVs or independent BIOS vendors), including AMI, Insyde, and Phoenix, use the reference code extensively. Project Mu, Microsoft's hardened EDK2 fork, used the code containing the vulnerable component as well.

The vulnerability, tracked as CEV-2026-97631, is a memory-corruption flaw in an ICMP component that attackers can exploit to execute malicious code that runs before the operating system and firmware can implement defenses like Secure Boot, which establishes a chain of trust for all code, including the Trusted Platform Module (TPM), loaded during boot.

https://arstechnica.com/security/2026/10/code-execution-flaw-in-uefi-firmware-threatens-most-servers-on-planet-earth/

# Measured Boot and the TPM




**Trusted Platform Module (TPM)**

a protected chip or processor component that holds keys, signs data, and keeps integrity registers

**Platform Configuration Register**

Cannot be set, only extended:

$$new = \text{hash}(old \parallel measurement)$$

A hash chain inside the chip.

**Secure Boot decides whether code may run. Measured boot records what ran. The TPM is the record keeper, not the inspector.**

# Using the Boot Measurements

### Local: unlocking the disk

The disk key is sealed to expected PCR values. If a change affects the measurements the policy covers, the TPM withholds the key and the disk stays encrypted. BitLocker works this way.

### Remote attestation

1. The service sends a fresh challenge
2. The TPM signs the PCR values with the challenge
3. The service checks the signature and compares the values with approved ones

### The signature is still valid

An old signed boot loader passes Secure Boot, but its hash differs, so the content fails the check

### Two limits

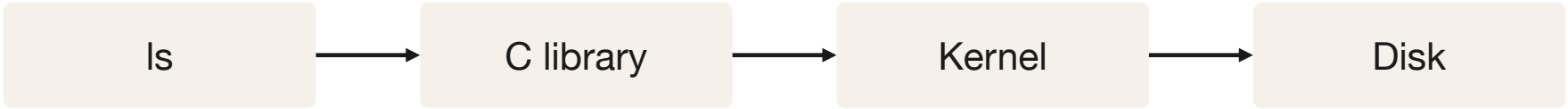
Each approved version needs its own expected values. A kernel exploit after boot leaves them unchanged.

**Checking the signature and comparing the measurements are separate steps. A valid signature alone approves nothing.**

# Concealing Malware with Rootkits

# Rootkits Change What the System Reports

A **rootkit** conceals unauthorized activity or access by altering system programs, libraries, or the kernel. It hides files, processes, connections, and configuration changes.




**user-mode rootkit: replaces the program or the library**

**kernel-mode rootkit: filters inside the kernel**

**User-mode rootkit**
A modified ls omits a file; a modified ps omits a process. LD_PRELOAD can substitute one library function for another. A trusted copy of the tool exposes the difference.

**Kernel-mode rootkit**
A driver filters the records every utility receives, so unmodified tools all show the same incomplete listing

# Story: Sony BMG's Copy-protection Rootkit

**2005**
Sony BMG, a record company, shipped audio CDs with copy-protection software. On Windows, it installed a kernel driver, aries.sys.

**What the driver did**
It intercepted kernel interfaces and removed every file, process, and registry entry whose name began with $sys$ from the results

**Anyone could use the hiding**
A backdoor named `$SYS$DRV.EXE` appeared within weeks, hidden by Sony's own rootkit

**The objects were still there**
Only the listings were filtered. A tool that read the disk directly could see them.

**A rootkit installed for a legitimate reason conceals whatever adopts its naming rule**

# Defending against Rootkits

**Cross-view detection**
compares what the system reports with what a different route shows. A file hidden from the directory API may still appear in the file system's own metadata.

**Inspect from outside**
Trusted recovery media reads the disk without starting the installed kernel. Memory evidence has to be captured before shutdown.

**Driver signing is not proof**
Stuxnet used drivers signed with certificates stolen from Realtek and JMicron. A signed driver can also have a vulnerability.

**Vulnerable-driver blocklist**
Microsoft blocks known vulnerable driver versions even when their signatures are valid

**A discrepancy needs investigation, not automatic classification: concurrent changes produce differences too**

# Malware on Mobile Devices

# Abusing Mobile Applications

Phones isolate each app and mediate access through permissions. Mobile malware usually misuses a granted capability rather than defeating the isolation.

**Trojanized application**
a useful program altered to include hidden behavior. Its useful purpose earns the permissions.

**Accessibility abuse**
An Android service that reads the screen and operates controls for users with disabilities can drive other apps

**System-level spyware**
An exploit chain escapes a messaging app's sandbox with no install or permission prompt (Pegasus, 2021)

**App-store review tries to reject such behavior before distribution; the sandbox still limits what else the app can reach**

# Triggers

# Triggers: When the Payload Acts

A **trigger** is the condition under which the payload activates: a date, a count of executions, a user action, or the presence of a particular target.

**Logic bomb**
code that acts when a specified condition occurs. It need not replicate and may sit inside a legitimate application.

**Stuxnet's trigger**
inspected the industrial controllers attached to the machine and activated its physical payload only on the configuration it targeted

**Immediate payloads**
A credential thief runs at startup. A worm's replication is itself the harm. Neither waits for anything.

**A trigger is the general idea; not every conditional branch in malware is a separate logic bomb**

# Communicating with an Operator

# Command and Control

**Command and control** (C2) is the exchange through which an operator directs compromised systems and receives status or results.

### Three jobs

1. Receive instructions
2. Fetch a later payload
3. Send stolen data out: **exfiltration**

### Hiding in HTTPS

An outbound web connection is allowed almost everywhere, and TLS hides the content

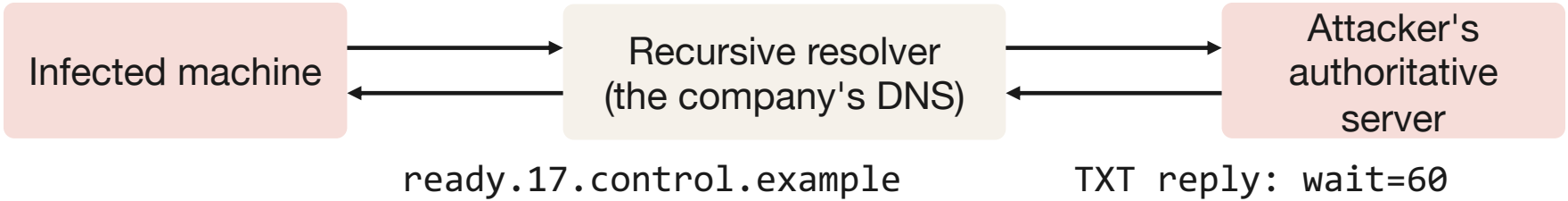
### Hiding in the cloud

Hosting and CDNs shared with legitimate services. Blocking a provider blocks business traffic too.

### Autonomous malware needs none of this

A worm replicates on embedded instructions, and ransomware can encrypt with an embedded public key

# Tunneling Data through DNS




**DNS tunneling**
embeds commands or data inside DNS queries and replies. The queried name carries data out; the reply record carries data in.

**Why it works**
The machine talks only to its permitted resolver, which relays the query through normal DNS to a domain the attacker controls

**Encoding makes data fit DNS fields; it does not hide it. Throughput is limited by message size and caching.**

# Finding Replacement Control Servers

**Dead-drop resolver**
a public page or account that publishes the current controller address. Updating the page redirects every infected machine.

**Domain generation algorithm**
malware and operator compute the same changing list of domains from a shared input, such as the date

**Peer-to-peer control**
infected machines relay instructions to each other, so removing one server leaves other paths working

**Lumma, 2025**
carried embedded address lists plus fallbacks hidden in Steam profiles and Telegram channels

**Conficker, 2009**
generated domains to find updates; Microsoft published the predicted list so defenders could block them

**A recovered algorithm lets defenders predict and block the rendezvous**

# Detecting and Disrupting Hidden Traffic

**What encryption leaves behind**

- connection timing and volume
- destination addresses
- queried names in resolver logs
- the process that opened the connection

**Beaconing**

repeated check-ins that announce availability or request work. Regular intervals stand out, so malware varies them.

**Sinkholing**

redirects traffic for malicious infrastructure to a defender's server. It interrupts contact and identifies victims; it removes nothing.

**Emotet 2021, QakBot 2023**

Investigators replaced the malware's update with their own file, or pushed an uninstaller. Other malware already installed remained.

**Malware that authenticates its commands with a key ignores a replacement server even after redirection**

# Payloads

# Stealing Credentials and Sessions

An infostealer collects what can be sold or reused: browser passwords, session tokens, wallet secrets, and documents. Moving it out of the environment is **exfiltration**.

**Bearer tokens**
A session token can be used by whoever holds it. A copied token reuses the session without the password or second factor, as long as the server accepts it.

**Multifactor protects the login**
not the session afterward. The browser has to hold usable secrets, and malware with the user's permissions can often reach the same interfaces.

**Removing the infostealer does not recall what was copied. Recovery means revoking sessions and replacing credentials.**

# Ransomware and Wipers

**Ransomware**
encrypts files with symmetric keys and locks those keys with an attacker-controlled public key. The private key stays with the attacker, so encryption needs no live connection.

**Wiper**
destroys data or discards what recovery would require. It may display a ransom demand anyway, as NotPetya did in June 2017.

**Steal first, then encrypt**
Restoring a backup repairs availability but does not recall the copies the attacker threatens to publish

**May 2017: WannaCry**
combined file encryption with worm propagation. Blocking the spread protected other machines but not the one already encrypting.

**Whether recovery is possible depends on the design, not on what the screen says**

# Remote Control, Botnets, and Sabotage

**Remote access trojan**
gives an operator file transfer, command execution, screen viewing, and further installation

**Botnet**
compromised machines under coordinated control: DDoS attacks, spam, or cryptocurrency mining

**Sabotage**
altering files, application state, or industrial control software while showing the operator misleading readings

**2016: Mirai**
logged in to routers, cameras, and video recorders with factory-default passwords and used them for DDoS attacks

**2010: Stuxnet**
changed Siemens controller code to manipulate centrifuge operation while concealing the changes from the operators

**For a legitimate tool, the question is who installed it and why**

# Propagation

# Propagation

A **worm** searches for reachable systems and attempts to make them execute another copy. Each copy carries the earlier stages with it: infection and delivery happen again on the next machine.

**Replication can be the damage**
SQL Slammer's scanning alone produced widespread network congestion, with no payload that touched a file

**The stage that breaks the sequence**
Code Red and Slammer spread before doing anything else. The Morris worm never had a payload beyond spreading.

**Blocking propagation protects the machines not yet infected**
and leaves the payload active on the ones that are, as WannaCry showed

# Malware Detection

# Signatures and Static Analysis

**Malware signature**
a pattern that recognizes malicious content: a file hash, a byte sequence, or a structural rule. A hash matches one exact file; a rule may match a family.

**Static analysis**
examines a program without running it: strings, imported functions, file structure, instructions. A static heuristic flags suspicious construction rather than one known pattern.

**Fast, and only for what is known**
Changing one byte defeats a hash match, but may leave a larger structural match intact

**Not a digital signature**
A malware signature recognizes content. A digital signature authenticates a publisher.

**Known malicious programs often share recognizable structure, which gives a quick decision before anything runs**

# Evading Static Detection

**Packing**
hides the original executable until runtime. The scanner sees the unpacker and encrypted data.

**Polymorphic malware**
keeps a stable body but encrypts it differently in each copy and varies the code that restores it

**Metamorphic malware**
rewrites its own instruction sequences into equivalent forms, so there is no fixed body to find

**What survives**
A known packer is itself recognizable. Memory after restoration holds the real instructions. And the behavior, theft or control, is the same in every variant.

**Detection targets surviving structure, the transformation mechanism, or the resulting behavior, not one fixed byte sequence**

# Behavioral Detection and EDR

Document app starts an unusual interpreter → Interpreter reads browser data → External upload begins

Each event alone is ordinary. The sequence, and its origin, is the evidence.

**Behavioral detection**
evaluates actions during execution: file access, process creation, memory changes, configuration changes, network connections

**Endpoint detection and response (EDR)**
records and correlates those events so the chain from download to script to connection can be reconstructed later

**Strict rules block legitimate administration (false positives); loose rules allow the same tools to be abused (false negatives)**

# Analysis Sandboxes and Their Evasion

**Analysis sandbox**
run the suspicious program in an isolated environment, such as a virtual machine, and record what it does

**Evasion**
The program inspects its environment, waits for mouse activity or a target configuration, or delays past the observation window, and does nothing

**A clean run proves little**
The payload server may be offline, the required document missing, or the program waiting for a command. Isolation contains; it does not classify.

**Longer observation and realistic conditions expose more, but a short run cannot show what the program might do later**

# Memory-resident Malware and Rootkits

**Antimalware Scan Interface (AMSI)**
lets Windows applications hand content to security software for scanning. An interpreter submits a script after decoding it, so the scanner sees what will actually run.

**What a scanner can still examine**
the script given to an interpreter, instructions in executable memory, the process that started the code, and its network activity

**When the kernel is lying**
A kernel rootkit alters reports or disables the monitor. An independent network record still shows that a connection happened, though not what it carried or which code sent it.

**Code that exists only in memory has no file to scan, but it still has a script, a process, and a connection**

# Reputation, Telemetry, and the Detector

**Reputation-based detection**
uses prior observations of a file, publisher, address, or domain. A new executable contacting known theft infrastructure deserves a look.

**Telemetry**
recorded operational data from many machines. Combined, it exposes rare process relationships and coordinated infrastructure invisible from one machine.

**Rare is not malicious**
A newly released legitimate program has few observations, and a familiar publisher does not authorize every use of its programs

**Protecting the detector**
Security software runs with broad privileges and parses hostile input. It is part of the trusted computing base, and a flaw in it exposes the machine.

**A compromised monitor does worse than miss: it corrupts the evidence administrators rely on**

# The End