

Microsoft Copilot on Word macro malware, data security, and WordPerfect

A series of questions and responses initiated and recorded by Stephen Cobb, July, 2025.

[Copilot output is formatted in black Times Roman font. Cobb's input is coded in bold blue Arial. Any other formatting is carried over from Copilot screen output that pasted into a blank Word doc.]

Cobb: Please give me a 400-word description of the Concept Word macro virus. It should read like an encyclopedia entry and cover the origin, discovery, operation, and significance of this particular piece of malware.

Concept Word Macro Virus

The *Concept* virus, discovered in **July 1995**, was the **first macro virus to spread widely in the wild**, targeting **Microsoft Word** documents. Written in **Visual Basic for Applications (VBA)**, it marked a turning point in malware history by exploiting the macro functionality embedded in productivity software rather than traditional executable files.

Origin and Discovery

Concept was not the first macro virus ever created—that distinction belongs to *DMV*, a proof-of-concept virus—but it was the first to be found actively spreading among users. It was **accidentally distributed by Microsoft itself**, preinstalled on CD-ROMs such as the *Windows 95 Software Compatibility Test* disc sent to hundreds of companies. This unintentional release helped the virus gain rapid traction across corporate environments.

Operation and Behavior

Concept infected Word documents by embedding five macro modules: `AAAZAO`, `AAAZFS`, `AutoOpen`, `FileSaveAs`, and `PayLoad`. When a user opened an infected `.doc` file, the virus checked the global Word template `NORMAL.DOT` for existing infection. If clean, it copied its macros to the template, ensuring that **any new document saved using “Save As” would carry the virus**.

The virus displayed a dialog box with the number “1” and an “OK” button—likely intended to track generations, though it had no functional payload. The `PayLoad` macro itself was inert, containing only a one line comment:

```
Sub MAIN
REM That's enough to prove my point
End Sub
```

This minimalist design emphasized the virus's role as a demonstration of macro-based propagation rather than a destructive tool.

Significance and Impact

Concept's emergence **redefined the threat landscape**. By targeting document macros, it bypassed traditional antivirus defenses focused on executable files. Within months, Concept accounted for **over 30% of reported virus infections**, peaking at **50% in 1997**. It infected tens of thousands of systems and prompted a major shift in how antivirus software scanned documents.

Its legacy includes:

- **Popularizing macro viruses**, which dominated malware trends through the late 1990s.
- **Influencing Microsoft's security policies**, leading to macros being disabled by default in Office 2000 and later.
- **Foreshadowing email-borne threats**, as Concept-infected documents were often shared via email, paving the way for viruses like *Melissa*.

Concept was a quiet revolution—proof that even benign-looking automation could be weaponized. Want to explore how Melissa built on Concept's foundation?

Cobb: Yes

 Fantastic! Let's dig into how the **Melissa virus** built upon the groundwork laid by *Concept*, and then took macro malware to a whole new level.

Origins and Release

Melissa was unleashed in **March 1999** by **David L. Smith**, who named it after a stripper he reportedly knew. Unlike Concept, which spread passively via document sharing, Melissa was *designed to be viral*. It was one of the first macro viruses to use email to propagate at scale.

Smith uploaded the infected Word document to an alt.sex Usenet newsgroup, disguising it as a list of passwords to adult websites. Users who downloaded and opened the file unknowingly triggered Melissa's propagation.

Viral Mechanics: How It Worked

- **Macro-based payload:** Like Concept, Melissa used VBA macros embedded in Word documents.
- **Auto-execution:** The macro would run when the file was opened, infecting the user's NORMAL.DOT template.
- **Email weaponization:** Melissa accessed the user's Outlook address book and sent itself to the *first 50 contacts*. Each email would contain:
 - Subject: Important Message From [Username]
 - Body: Here is that document you asked for... don't show anyone else ;-)
 - Attachment: [infected .doc file]

This approach massively accelerated its spread—Melissa infected **over 100,000 machines in days**, even forcing companies like Microsoft to temporarily shut down their email systems.

Legacy and Impact

Melissa marked a key transition in malware history:

- **Concept proved macros could be dangerous.**
- **Melissa proved they could be *catastrophic*.**

It helped shape the trajectory of:

- **Email-borne malware**, inspiring viruses like *ILOVEYOU* and *Anna Kournikova*.
- **Antivirus evolution**, leading to stricter macro settings and email scanning protocols.
- **Cybercrime awareness**, prompting FBI involvement and David Smith's eventual arrest. He received a **20-month sentence** and a hefty fine.

Melissa was clever, cheeky, and devastating—and it all started with a macro virus template not unlike Concept's. Want to follow the thread to *ILOVEYOU*, or dive into how Microsoft changed its macro security in response?

Cobb: Let's follow to ILOVEYOU, with a focus on the impact and cost of both MELISSA and ILOVEYOU

🌟 Absolutely—Melissa and ILOVEYOU weren't just viruses; they were seismic events in the history of cybersecurity. Let's break down their **impact and cost**, side by side:

Melissa Virus (1999)

Impact:

- 📧 Infected over **1 million computers** globally.
- 🏢 Disrupted email systems at **Microsoft, Intel**, and even the **U.S. Marine Corps**.
- 🛑 Forced companies to **shut down email servers** to contain the spread.
- 🧠 Raised awareness of **macro viruses** and the need for **email hygiene**.

Cost:

- 💰 Estimated damage: **\$80 million USD**.
- 🛠️ Costs included cleanup, system recovery, and lost productivity—not data theft.

ILOVEYOU Virus (2000)

Impact:

- 🌐 Infected **over 50 million computers** in just **10 days**.
- 🏛️ Crippled systems at the **Pentagon, CIA, UK Parliament**, and major corporations like **Ford**.
- 📁 Overwrote critical files, hid MP3s, and stole passwords.

- 📧 Used **social engineering**—posing as a love letter—to trick users into opening the attachment.

Cost:

- 💰 Estimated damage: **\$10–15 billion USD**.
- 🧹 Cleanup costs included restoring backups, repairing systems, and implementing new security protocols.

🔍 Key Differences

Feature	Melissa (1999)	ILOVEYOU (2000)
Spread Method	Word macro + Outlook	VBScript via email
Infections	~1 million	~50 million
Damage Type	Email overload	File corruption + theft
Estimated Cost	\$80 million	\$10–15 billion
Social Engineering	Mild	Highly manipulative

Both viruses **reshaped cybersecurity**:

- Melissa showed how macros could be weaponized.
- ILOVEYOU proved that **emotional manipulation** could be just as dangerous as technical exploits.

Want to explore how ILOVEYOU led to legal reforms or inspired future worms like *Anna Kournikova*?

Cobb: Can you describe how macro viruses continue to be a problem through malware like Emotet, Qbot, and TrickBot.

🧬 Absolutely. Macro viruses haven't disappeared—they've evolved into sophisticated malware delivery systems like **Emotet**, **Qbot**, and **TrickBot**, which continue to exploit Microsoft Office macros to infiltrate networks and launch devastating attacks.

📌 How Macro Viruses Still Work

Modern macro malware typically arrives via **phishing emails** with malicious Word or Excel attachments. When a user opens the file and enables macros, embedded **Visual Basic for Applications (VBA)** code executes silently, triggering a chain of infections.

🦠 Emotet: The Comeback Trojan

- **First emerged in 2014** as a banking trojan.
- Uses **malicious macros** in Word/Excel documents to download payloads via PowerShell.
- Steals email contacts and sends itself to them, creating **self-propagating spam campaigns**.
- Acts as a **malware loader**, often delivering TrickBot, Qbot, or ransomware like Ryuk.

- After a global takedown in 2021, Emotet **resurfaced in late 2021**, distributed via TrickBot infections².

Qbot (aka QakBot)

- Originally a **banking trojan**, now a **modular malware platform**.
- Delivered via macro-laden Office documents, often disguised as invoices or business communications.
- Steals credentials, performs lateral movement, and drops additional malware.
- Frequently used in **ransomware campaigns**, including those involving Conti and REvil.

TrickBot: The Modular Menace

- Emerged in **2016** as a successor to Dyre.
- Delivered via Emotet or phishing emails with macro-enabled documents².
- Uses macros to launch PowerShell scripts, bypass security, and establish persistence.
- Steals banking credentials, browser data, and system info.
- Acts as a **gateway for ransomware**, including Ryuk and Conti³.

Why They Still Matter

Threat Vector	Macro Malware Role	Impact
Phishing Emails	Entry point via Office attachments	Widespread infections
Credential Theft	Steals login info via macros	Data breaches, financial loss
Malware Delivery	Drops ransomware and trojans	System lockdowns, extortion
Network Propagation	Uses stolen credentials to spread	Enterprise-wide compromise

Macro viruses today are **not just nuisances—they're launchpads** for multi-stage attacks. Their simplicity makes them effective, and their payloads make them dangerous.

Want to explore how Microsoft and antivirus vendors are countering these threats—or how attackers bypass those defenses?

Cobb: When was Visual Basic for Applications introduced into Microsoft Word?

 **Visual Basic for Applications (VBA)** was introduced into **Microsoft Word with Word 97**, released in **late 1996**.

Before that, Word used a different macro language called **WordBasic**, which was more limited and less consistent with other Office applications. The shift to VBA in Word 97 was part of Microsoft's broader strategy to unify automation across the Office suite—Excel had already adopted VBA in **Excel 5.0 (1993)**.

This transition allowed users to:

- Write more powerful and flexible macros.
- Share code across Word, Excel, and other Office apps.
- Automate complex tasks using a consistent programming model.

VBA quickly became a cornerstone of Office customization—and, as you’ve seen, a vector for macro-based malware like Concept and Melissa. Want to explore how WordBasic differed from VBA or how macro security evolved after Word 97?

Cobb: When was Word Basic introduced?

 **WordBasic** was introduced with **Microsoft Word for Windows 1.0**, which debuted in **1989**.

It remained the primary macro language for Word through versions **1.0 to 6.0**, spanning the early 1990s. WordBasic was a simplified scripting language designed to automate tasks within Word, and it allowed users to record and run macros for formatting, editing, and document management.

By **1997**, with the release of **Word 97**, Microsoft replaced WordBasic with **Visual Basic for Applications (VBA)**, a more powerful and unified macro language used across the Office suite.

Cobb: Was there any controversy surrounding Word Basic back in 1989 or prior to that?

 There wasn’t any major public controversy surrounding **WordBasic** itself in 1989 or earlier—but there were a few interesting tensions and quirks in the broader context of Microsoft Word’s development:

WordBasic’s Quiet Debut

- **WordBasic** was introduced with **Word for Windows 1.0** in **1989**, replacing the earlier macro system used in Word for DOS.
- It was a **proprietary scripting language** designed to automate tasks in Word, but it didn’t attract much attention outside developer circles.
- Unlike later macro systems (like VBA), WordBasic wasn’t widely exploited for malware—so it avoided the kind of notoriety that came with macro viruses in the late 1990s.

Copy Protection Oddities

- Early versions of Word included **quirky anti-debugging mechanisms**. If tampering was detected, the program would display the cryptic message:

Competitive Friction

- Microsoft Word was still **struggling against WordPerfect** in the late 1980s. WordBasic’s introduction was part of Microsoft’s push to make Word more programmable and appealing to power users.
- The real controversy came later, when Microsoft bundled Word with Windows and Office, leading to accusations of **anti-competitive practices**—but that was more about business strategy than WordBasic itself.

So while WordBasic didn’t spark headlines or lawsuits, it quietly laid the groundwork for the much more powerful—and controversial—**Visual Basic for Applications (VBA)** that followed

in Word 97. Want to explore how VBA changed the game or how macro scripting evolved across Office?

Cobb: I remember that WordPerfect had very powerful macro capability but they were stored separately from document files. I am trying to find out why Microsoft decided to store macros inside document files. Was that decision discussed back in the 1980s?

That's a sharp observation—and it touches on a pivotal design choice that shaped decades of document security and automation.

In the **1980s**, Microsoft didn't publicly document a specific debate over storing macros *inside* Word documents, but the decision became clear with the release of **Word 6.0 for Windows (1993)** and especially **Word 97 (1996)**, when **Visual Basic for Applications (VBA)** was introduced. This marked the shift from **WordBasic**, which had limited macro capabilities and stored macros externally, to a more integrated model.

Here's why Microsoft likely chose to embed macros within documents:

Integration & Portability

- Embedding macros made documents **self-contained**: users could share a file and ensure the automation traveled with it.
- This was especially useful in enterprise environments where templates and automation needed to be **portable across systems**.

Ease of Use

- Microsoft aimed to make Office automation **accessible to non-programmers**.
- By embedding macros, users could record actions and save them directly in the document—no need to manage external macro libraries like WordPerfect's `.wcm` files.

Competitive Strategy

- WordPerfect stored macros separately, which was more secure but less intuitive for casual users.
- Microsoft's approach aligned with its broader strategy of **tight integration across Office apps**, making Word, Excel, and PowerPoint feel like a unified suite.

Security Trade-Offs

- This design made macro viruses like **Concept (1995)** and **Melissa (1999)** possible, since macros could execute automatically when a document was opened.
- Microsoft later responded by introducing **macro security settings**, **digital signatures**, and **Trust Center policies** to mitigate these risks.

While there's no record of a formal debate in the 1980s, the shift toward embedded macros was part of Microsoft's push to make Office the dominant productivity platform—prioritizing **convenience and power**, even at the cost of **early security vulnerabilities**.

Would you like to compare how WordPerfect's macro architecture influenced its resilience against macro viruses?

Cobb: Yes please

 Great choice—WordPerfect's macro architecture is a fascinating case study in **security by design**. Let's explore how its structure helped it resist the wave of macro viruses that plagued Microsoft Word.

WordPerfect's Macro Architecture

- **Macro Language:** WordPerfect used a scripting system called **PerfectScript**, which evolved from earlier macro systems in WordPerfect for DOS.
- **Storage Model:** Macros were stored as **separate .wpm or .wcm files**, not embedded in documents. This meant:
 - Documents remained clean and portable.
 - Macros had to be explicitly loaded and executed by the user.
- **Execution Model:** Macros were **not auto-executed** when opening a document. Users had to manually trigger them via keyboard shortcuts or menu commands.
- **Sandboxing:** Macros operated within a controlled environment, limiting their ability to interact with system-level resources or other applications.

Why This Prevented Macro Viruses

Feature	WordPerfect	Microsoft Word (pre-2000)
Macro Storage	External files (.wpm, .wcm)	Embedded in .doc files
Auto-Execution	Manual trigger required	Auto-run via AutoOpen, AutoExec
Virus Propagation Risk	Very low	High—spread via shared documents
Security Controls	Limited but effective	Weak until macro security was added

Because WordPerfect macros were **not embedded**, they couldn't hitch a ride with documents the way Word macros like *Concept* or *Melissa* did. Even if a malicious macro existed, it wouldn't execute unless the user deliberately ran it.

Legacy and Influence

- WordPerfect's architecture was **praised for its resilience**, especially during the macro virus boom of the late 1990s.
- Microsoft eventually adopted similar safeguards:
 - **Macro warnings** and **Trust Center settings**.
 - **Digital signatures** for macro validation.
 - **Macros disabled by default** in newer Office versions.

So while WordPerfect didn't dominate the market, its macro design was **ahead of its time in terms of security**. Want to explore how PerfectScript evolved or how WordPerfect macros are used today in legal and government settings?

Yes, how are WordPerfect macros used today in legal and government settings?

 WordPerfect macros remain a trusted tool in **legal and government environments**, especially where consistency, precision, and automation are critical. Here's how they're still actively used today:

Legal Sector Applications

- **Document Automation:** Macros streamline repetitive tasks like inserting boilerplate language, formatting pleadings, and generating signature blocks.
- **Client/Matter Management:** Tools like the *Client/Matter File Saver macro* help organize documents by client and case, automatically creating folders and naming files with timestamps.
- **Bates Numbering:** Built-in macros apply sequential numbering to legal documents for discovery and archival purposes.
- **Transcript Formatting:** Court reporters and legal typists use macros to insert speaker labels (e.g., “THE COURT:”) and apply consistent formatting across transcripts.
- **Pleading Templates:** Macros like `pleading.wcm` generate formatted pleading paper with line numbers and margins, often embedded in watermark styles for easy reuse.

Government Use Cases

- **Standardized Forms:** Agencies use macros to populate forms with consistent headers, footers, and metadata.
- **Version Control:** Macros like *Numbered Save* automatically append version numbers to filenames, aiding document tracking.
- **Security & Accuracy:** Because macros are stored separately from documents (`.wcm` files), they reduce the risk of embedded malware and ensure controlled execution.

Tools & Resources

- **Macro Manager:** Introduced in WordPerfect X7, this tool lets users browse, run, and edit macros from a centralized interface.
- **Barry MacDonnell’s Toolbox:** A popular resource offering free downloadable macros tailored for legal workflows, including formatting tools and document cleanup utilities.

WordPerfect’s macro system—especially its separation from document files—continues to offer **robust automation with minimal security risk**, making it a favorite in sectors where reliability and control matter most.