The Malware Problem

Michael Stephen Fiske

Aemea Institute, San Francisco, California, U.S.A.
mf@aemea.org

Abstract

The difficulties in protecting existing systems from malware and/or detecting the existence of malware is best understood from a computer architecture perspective. A solution to the malware problem requires new concepts in mathematics and computer science to better explain the computer security weaknesses that have enabled recent cyberattacks.

1 What is the nature of the problem?

The difficulties in protecting existing systems from malware and/or detecting the existence of malware is best understood from a computer architecture perspective. A solution to the malware problem requires new concepts in mathematics and computer science to better explain the computer security weaknesses that have enabled recent cyberattacks.

2 Why is this important?

Malware can exploit a fundamental cybersecurity weakness in current computers: user authentication (identity) is not securely bound to the authorization and execution of an action. Strong biometric and cryptographic authentication can be circumvented when malware is able to hijack the machine (computer) that executes these authentication and authorization operations. In practical terms, this means financial transactions can be breached or hijacked by malware even when the transaction system uses strong cryptography and identity authentication.

3 Limitations of current cybersecurity methods

Some approaches attempt to conceal and protect a computation by using a physical or virtual barrier (e.g., firewall or private network). These approaches are not successful. Mobile devices and internet connectivity enable malware to circumvent these boundaries. Cryptographic approaches often assume $P \neq NP$ and use an NP-problem to provide confidentiality <http://www.claymath.org/millennium/P_vs_NP/pvsnp.pdf>. Homomorphic cryptography <http://crypto.stanford.edu/craig/> executes operations that are twelve orders of magnitude too slow. If the execution is tampered with, then this destroys the computation even though the adversary may not successfully decrypt it. Homomorphic cryptography executing on a register machine is still susceptible to a fundamental weakness discussed below. The von-Neumann architecture has a stored-program digital computer that uses a CPU and separate memory to store instructions and data. Generally, only a single instruction is executed in sequential order and time is not used in the instructions.

Some current cybersecurity approaches use evolution of programs on a von-Neumann architecture <http://bit.ly/KdkTiR>. Some approaches use obfuscated code <http://bit.ly/KwOPTU>. Some approaches build malware detection software. In 1987, Dr. Fred Cohen <http://bit.ly/KdmVj9> proved that there is no algorithm that can perfectly detect all malware <http://all.net/books/Dissertation.pdf>. The severity of recent cyberattacks demonstrates the limitations of the malware detection approach. Over the last 20 years, DARPAs CRASH program <http://bit.ly/Q3QTa8> compared the number of lines of source code in security software versus malware. Security code grew from about 10,000 to 10 million lines; while malware was almost constant at 125 lines. An almost constant number of lines of malware code are needed to hijack a register machine program: independent of the computer programs size and independent of the complexity of the security software protecting the program.

4 Why is the register machine architecture vulnerable?

Current approaches rely on operating systems that execute on a register machine architecture. From a mathematical perspective, register machine programs execute computational steps that are topologically disconnected: This mathematical property of register machines creates hijacking opportunities for malware. The sequential execution of instructions in a register machine program make it susceptible to hijacking and sabotage. By inserting only one *jmp WVCTF* instruction into the register program or changing one legitimate jmp instruction to *WVCTF*, the program is hijacked.

Malware Instructions (polymorphic variant)

WVCTF:		mov	eax,	drl
		jmp	Loc1	
Loc2:		mov	edi,	[eax]
LOWVCTF:		pop	ecx	
		jecxz	SFMM	
		inc	eax	
		mov	esi,	ecx
		dec	eax	
		nop		
		mov	eax,	0d601h
		jmp	Loc3	
Loc1:		mov	ebx,	[eax+10h]
		jmp	Loc2	
Loc3:		pop	edx	
		pop	ecx	
		nop		
		call	edi	
		jmp	LOWVCTF	
SFMM:		pop	ebx	
		Pop	eax	
		stc		

After the register machine program has been hijacked, even if there is a friendly routine to check if the program is behaving properly, *this safeguard routine will never get executed. The sequential execution of single register machine instructions cripples the program from defending and repairing itself.*

5 Design Strategy

In the short term, a malware containment strategy is necessary to adequately address cybersecurity weaknesses in current chip architectures. In the long term, a new computing machine is needed to execute malware resistant programs. See <http://www.aemea.org/Turing100>.