Security and Computer Architecture

R_Ismeet Kaur Makkar
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
ismeettkaur@gmail.com

ABSTRACT

Information Security is the process of protecting data and information on any device, which has a processor and memory, from unintended and unauthorized access, use, disclosure, revelation, modification, or destruction as well as over all the private and public networks including the Internet. Security deals with three major areas: confidentiality, integrity and availability. It is imperative to design secured software given the growth in importance and the increase in reliance on computer systems, tablets and mobile devices throughout the world. Design for security is an essential aspect of the design of future computers. However, security is not well understood by the computer architecture community. Many important security aspects have evolved over the last several decades in the cryptography, operating systems, and networking communities. It is essential for computer science and security professionals to understand both hardware and software security solutions. We will briefly talk about hardware support for: buffer-overflow prevention, secure information processing, cryptography, dynamic information flow tracking, tamper resistant and verified software, address-space randomization to prevent code injection, hardware-based virus and intrusion detection, secure cryptographic co-processors, and various "trusted" computing initiatives.

1. INTRODUCTION

Securing important information and implementing means to ensure that the information is not breached has been given prime importance since early ages. The need to protect stored data and its' transmission has become all the more important considering the growth and widespread use of electronic devices, computer systems, tablets and mobile devices, for data generation and processing and all business being conducted on Internet. A majority of organizations have implemented or are in the process of implementing security improvements to their systems, according to Symantec. Computer Security has many definitions, the most common being "preservation of confidentiality, integrity and availability of information." It is essential to understand the three terms individually before we dig deeper into security.

Confidentiality deals with preventing unauthorized reading of information. A common example would be bank customers who would not want their account information be read by anybody. Integrity deals with detecting and preventing unauthorized writing of data. An example again would be bank which does not want the customer to alter their account balances. Data availability has of-late been a fundamental issue. With the businesses being conducted on the Internet globally, it is the responsibility of the vendors that data is always available. Denial of service attacks on a few vendors every now and then, cut down the access to information.

Going beyond CIA, Information security comprises of four components: cryptography, access control, protocols, and software. Cryptography makes up the techniques to conceal data in such a way that only the people authorized or users who have the key can read/alter the data. Access control deals with authentication - whether or not a user has access to information, and authorization – what are the privileges of the user in accessing the information. Protocols define how to securely transfer information from one end to another ensuring confidentiality and integrity. It is rightly said that a system is no more secure than the human responsible for operating the computer system. After that the easiest way to compromise the security of the system is by exploiting the bugs in the software or introduction of bugged software in the system using external media. The most common form of attack exploiting the software has been buffer overflow.

2. BUFFER OVERFLOW

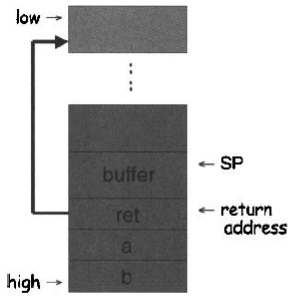
Buffer overflow is an unusual condition when a software program writes excessive data onto the memory buffer thereby overrunning the buffer's boundary and overwriting the data in the adjacent memory. The overflow can occur due to insufficient check conditions on the size of data to be written or copied to the destination. The exploit or security breach occurs when the overflowing data corrupts the data in the adjacent memory addresses. Buffer overflow is the most common vulnerability that is used by the attackers to execute malicious code on the target system, gain access to unauthorized sections on the system, perform incorrect operations using the software or bring the system down. Buffer overflow attacks have been known for 25 years yet they still pose the biggest threat to all software.[3]

2.1 Buffer Overflow - Existing Solutions

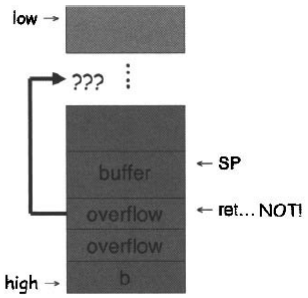
There are a number of solutions to guard against code corruption by untrusted data but all the solutions have one problem or another.

Canary-based buffer overflow protection uses random known values that are placed between a buffer and control data on the stack to monitor buffer overflows. Canaries are placed before the beginning of protected data and the value of the canary is verified each time protected data is used. The value of canary will change when data in the protective memory bound is overwritten by

buffer overflow attack, and thus canary will provide a check for buffer overflow detection. Canary has been implemented in software using stack linking information and heap chunk metadata, and hardware implementations have been proposed to protect the stack return address and work with unmodified binaries.[1]



Stack Example



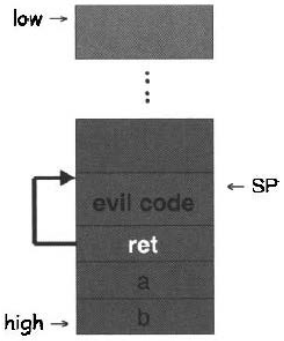
Buffer Overflow Causes a Problem

StackGhost is a hardware based implementation for buffer overflow protection. It implements a XOR based encryption of the return address. It was invented by Mike Frantzen and uses a special hardware feature of SPARC architecture to detect modifications to the return address pointers and protect the software application without adding any additional check. It is a simple tweak to the windows register fill and spill functions which make it difficult for the attackers to exploit buffer overflows. There is some performance impact which is being optimized and integrated continuously.[3b]

However, buffer overflows may be exploited without overwriting canary values in many situations. For example, in a system with stack canaries, buffer overflows may overwrite local variables, even function pointers, because the stack canary only protects stack linking information. Similarly, heap overflows can overwrite neighboring variables in the same heap chunk without overwriting canaries. Additionally, this technique may change data structure layout by inserting canary words, breaking compatibility with legacy applications. Canary-based approaches also do not protect other memory regions such as the global data segment, BSS or custom heap allocation arenas.

Non-executable data protection prevents stack or heap data from being executed as code. Some hardware (and many operating systems) support this no execute, or NX bit. Modern hardware platforms, including the x86, support this technique by enforcing executable permissions on a per-page basis. Data Execution Prevention (DEP) is well known among software developers when writing code to allow code to be executed only in the memory areas marked as executable. Intel markets the feature as the XD bit, for eXecute Disable. AMD uses the marketing term Enhanced Virus Protection and the ARM architecture refers to the feature as XN for eXecute Never. This approach, however, only prevents buffer overflow exploits that rely on code injection. It is not essential for an attacker to inject code to take control. Attackers can take control of an application by using existing code in the application or libraries which can be just as powerful as a code injection attack.[1]

The attackers can defeat canary defense by trying to execute already existing piece of code in the codebase. This is called return-to-libc attack and the attacker must be able to locate the code to be executed, while other attackers trying to execute shellcode injected on the stack have to find the stack first.



Return-to-libc attack

Address space layout randomization (ASLR) is a buffer overflow defense that randomizes the memory locations of system such that the key areas of the program are loaded and arranged randomly. In a system with ASLR, the base address of each memory region, i.e. stack, executable, libraries, and heap, is randomized at startup. A buffer overflow attack will not work properly as it is not easy to locate the security-critical information in memory. ASLR has been adopted on both Linux and Windows platforms. However, ASLR is not backwards compatible with legacy code, as it requires programs to be recompiled into position-independent executables and will break code that makes assumptions about memory layout. ASLR has to be switched off or disabled for the complete application and dependent processes if ASLR is not supported by the executable or any shared libraries. Some real-world exploits such as the Macromedia Flash buffer overflow attack on Windows Vista bypassed ASLR because the vulnerable application or its third-party dependent libraries did not have ASLR support. Attackers can easily elude ASLR on x86-based systems using brute-force techniques. ASLR implementations can be compromised if pointer values are leaked to the attacker by techniques such as format string attacks.

2.2 Hardware Solutions

There is a lot of research done for an effective hardware solution against buffer overflow. To mention a few:

1. Lee et. al. and Ozodganoglu et. al. independently presented a method of protecting the return address stack by copying the return address onto another hardware managed stack. This stack has backing store in a protected area of memory. When the function returns from a call, the processor checks to make sure that the return value on the two stacks match.

2. HSAP employs a stack smashing protection which does not allow variables passed into other functions to be
allocated on the stack. HSDefender also employs similar mechanism as HSAP. They use XOR encrypted return address in an extra register.

3. XOMOS [6] is proposed to solve the buffer overflow problem through the use of signed memory. It is based on the theory that the complete code must be divided into modules and all the modules that need to be secured from code execution will have a key. Each module that needs to be secured is copied to the memory and is cryptographically signed using a key and then written back for storage. Untrusted applications or parts of code are given a different key and a separate piece of memory to execute with. If the untrusted code attempts to overwrite memory which it does not have privileges to write to, it will be unable to create a valid corresponding signature. When the memory is used, the incorrect signature will be detected and the code will not be executed. This approach could be used to provide protection against buffer overflows, but requires that the users rewrite their application completely by partitioning the code that could have a buffer overflow.

4. PointGuard is a compiler-based buffer overflow defense technique where the pointer and addresses are encoded using XOR operation, thus obfuscating the pointer and return address. This prevents write-based buffer overflow attacks but is not effective on read-based attacks. Microsoft did not release PointGuard but implemented an API based on similar functionality for Windows XP SP2 and Windows Server 2003 SP1 operating systems.[3,7]

5. Similar address protection techniques, called SecureBit and SecureBit2 have been developed by Krerk Piromsopa and Richard J. Enbody where the return address is checked against a redundant copy of the address guarded by a bit or word. It is based on the principle that it is necessary to preserve the integrity of the address across domain to prevent buffer overflow attacks.[7]

Intel's Trusted Execution Technology (Intel TXT) uses cryptographic techniques and Trusted Platform Module (TPM) to provide trust measurements so that local and remotely managed applications can use these measurements for making trust decisions. This technology is aimed at defending the systems against any form of software based attacks which steal information from the system by injecting code, modifying the BIOS code or changing the platform configuration. Intel's new generation processors, Intel® Core™ vPro™ processor family, the Intel® Xeon® processor E5-2600, E5-1600, and E3-1200 product families, are all based on Intel TXT. [8]

3. SECURE INFORMATION PROCESSING

With the increase in amount of information available and the number of devices consuming that information it is important that the information be processed in a fast and secure manner. It holds such importance that there are government standards for secure information processing. FIPS (Federal Information Processing Standards) are a set of standards that describe document processing, encryption algorithms and other information technology standards for use within non-military government agencies and by government contractors and vendors who work with the agencies.
We will discuss some cases which have been developed for secure information processing.

3.1 Secure Coprocessors

Secure coprocessors provide a trusted environment for data storage and high computation to enable secured applications which require such conditions. But the technology in practice requires very high performing devices that are able to perform general computations along with specialized computations for a specific application in a secured and practical manner. This technology lacks the same thing that most computing applications lack – trustworthiness. The attackers can have access to the hardware, software, and data at its storage including the cryptographic keys and algorithms. A secure platform for building high-performance programmable secure coprocessor has been an area of research for a long time. The qualities that the platform must possess are: Tamper resistance, that is physical security against tamper attacks, fast speeds, that is high-speed cryptographic performance, general-purpose programmability which allows for programmers to easily develop and deploy secure software for these devices, and a security architecture that puts all this together securely. Sean W. Smith and Steve Weingart of IBM Secure Systems and Smart Cards developed a FIPS Level 4 Secure coprocessor architecture which is immune to physical attack so that it cannot reveal any internal hardware secrets. The architecture is based on two primary principles: tamper detection which is ensuring that the device detects all attacks, and tamper response which means ensuring that the device responds to any form of attacks that are detected by zeroizing any internal secrets before they are exploited or exposed.[9]

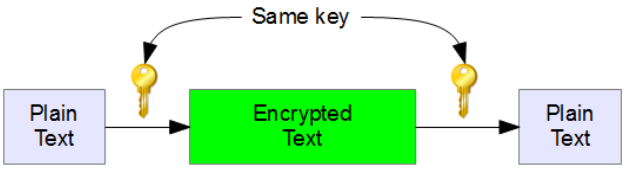
3.2 Secure processing on Mobile

Mobile devices make up a large portion of the devices connected on the Internet that generate data and process the information for a lot of businesses. This has been possible because of the smart mobile platforms like Android and Apple iOS. The new platforms pose a yet another security risk if these devices are being used for business critical data like enterprise emails or business reports. There are security mechanisms that have been implemented in these mobile platforms but a lot of security problems are reported on regular basis. This has led to a belief that a common and trust-worthy mobile platform has to be built for secure information processing especially for enterprises.

Some researchers have come together and created a prototype for mobile platform based on Android platform to combat typical attack scenarios. This architecture is based on Trusted Computing technology that implements consistent behavior of the computing systems. The consistent behavior is ensured by performing integrity checks on the system and then discovering any unexpected or unwanted characteristics. The basis of the technology is the integrity of the system and thus is implemented in hardware, as it is difficult to attack than corresponding software implementation. The researchers have come up with a hardware component called Trusted Platform Module (TPM) which implements a lot of reliable modules in the mobile platform and incorporates strong cryptographic functions as well as a true random number generator instead of pseudo random ones. They call this Secure mobile business information processing.[10]

4. CRYPTOGRAPHY

Cryptography comes from a Greek word meaning the art of writing secrets. Security is achieved using cryptography by transforming the given data into unintelligible form using a secret key, this means that the data is converted to a form which cannot be understood by anyone unless the reverse transformation is applied using the same/corresponding secret key. This process ensures that the information is retrieved and accessible only to the users which are meant to have access to the information. The first transformation when a key is applied to the plain text/data is called encryption and it produces cipher text that is intangible. This intangible data, cipher text can be converted back to readable information, plain text, by a process called decryption using the corresponding key. There are a number of algorithms that are designed to provide protection to sensitive information on the system where it is stored or when it is in transit over networks.

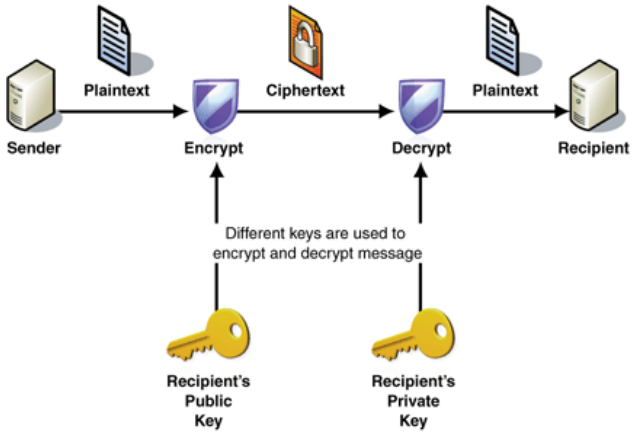


Private Key Cryptosystem

Cryptographic algorithms are either based on private-keys or public-key systems. Symmetric key ciphers (or private-key ciphers) use a shared secret key between the sender and receiver to encrypt as well as decrypt the information. AES is a well-known crypto algorithm that uses symmetric key. Asymmetric key cipher systems (or public-key ciphers) require two keys for each user – a private key that is kept secret and a public key which is made public to all. Both the keys are used to perform encryption and decryption based on the algorithm used. An example of such algorithm is RSA where public key of the recipient is used for encryption of data and receiver uses his private key to decrypt the cipher.

Having two keys is useful because the secret-key need not be shared between the users before starting the communication as is required in private-key cipher systems. But public key cipher systems take a longer processing time. They usually involve calculations using 1024-bit precision numbers which are computationally expensive, by as much as a factor of 1000, than equivalent private-key ciphers. Thus, typical cryptographic protocols like Secure Sockets Layer (SSL) use a public key algorithm to exchange the private-key among the communicating parties and then use the private key to share data amongst themselves. SSL is a standard secure protocol that provides secure communication between web servers and web clients. It is supported by most popular web browsers. Usually private-key has a faster processing speed which is the key to achieving fast response times except when the session is very short where public keys can be used.

Cryptographic algorithms provide the basis of securing information and hence need to be strong and have to be protected against tampering. The algorithms can be implemented in software, hardware, or a combination of both. The lowest cost solution is software implementation but is most prone to errors and exploits. Hardware implementation on the other hand is comparatively secured. An example of the hardware-only approach is the IDEA engine which implements and executes IDEA cipher efficiently with high performance output. However the problem with hardware approach is that the hardware is not flexible to be used for any other cryptographic algorithm. A team of researchers, Lisa Wu, Chris Weaver, and Todd Austin, have developed a high performance flexible architecture for secure communication. CryptoManiac is a fast and flexible cryptographic co-processor and addresses the primary bottleneck of efficiency in cipher systems through the application of an efficient VLIW architecture.[11]



Public Key Cryptosystem

Another type of hardware focused on secured cryptosystem is Hardware Security Module (HSM). This is a dedicated piece of crypto processor which has been designed specifically for the protection of crypto algorithms and ensuring a longer crypto key life cycle. The hardware security Modules have been developed by Safenet, trusted name in security and act as trusted entities that protect the cryptographic infrastructure of the users by processing, managing, and storing the keys of cryptographic algorithms inside a hardened, tamper-resistant device securely. HSMs provide protection for transactions, identities, and applications by securing cryptographic keys and provisioning encryption, decryption, authentication, and digital signing services for a wide range of applications.[12]

In general a cryptographic hardware component can be:

- a separate processor integrated into the CPU as a special purpose processor
- integrated in a coprocessor on a circuit board
- incorporated on a chip as an extension to circuit board which can be connected to the mainboard using a BUS interface like PCI, etc.
- an instruction set architecture like AES instruction set which is an integral part of the CPU

The most common use of cryptographic algorithms is in smart cards. The smart cards have a chip which ensures security by

performing check on the cardholder credentials (typically a PIN code) and performing cryptographic operations. The cards initially had a simple architecture and were embedded with the algorithm, which provided a few services but no cryptographic computation, and an 8-bit core, running at few tens of megahertz. Thus cryptographic hardware blocks had to be added to the chip. This helps the data processing to be faster than if software cryptographic modules are used. The disadvantage of using the hardware based implementation approach is higher cost of implementation than software based implementation. Maxim USIP PRO is an example of a secure single smart chip, which runs software implementations of the most common cryptographic algorithms (Table 1) very efficiently.[13]

Table 1. USIP PRO Cryptographic Performance

Algorithms	Speed
SHA-1	2083kBps
RSA 2048 CRT decrypt	400ms
RSA 2048 encrypt	18ms
ECDSA P-192 sign	23ms
ECDSA B-163 sign	16ms

5. DYNAMIC INFORMATION FLOW TRACKING

A wide range of security attacks from memory corruptions to SQL injections can be prevented using Dynamic Information Flow Tracking (DIFT). DIFT architecture associates a tag with every memory byte and word. This tag is used to mark the data that comes from sources not trusted by the system. These tags are carried forward in most operating systems when operations are performed on memory locations with these tags. This means that the tags are propagated from source to destination operands on performing operations. If tagged data memory is used in any unsafe way such as execution of an SQL command or dereferencing a pointer that has been tagged, a security exception is raised. Dynamic information flow tracking uses a simple hardware mechanism to track spurious information flows at run-time. On every operation the processor determines whether the result is spurious or not based on the inputs and the type of the operation. With the tracked information flows, the processor can easily check whether an instruction or a branch target is spurious or not, which prevents changes of control flows by potentially malicious inputs and dynamic data generated from them.

DIFT has several advantages as a security mechanism. DIFT analysis can be used on unmodified binaries. DIFT using hardware implementation has very less overhead and operates correctly with all types of legacy applications, even those with multithreading and self-modifying code. DIFT can potentially provide a solution to the buffer overflow problem that protects all pointers (code and data), has no false positives, requires no source code access, and works with unmodified legacy binaries and even the operating system.

A group of researchers [14] have worked on DIFT on Linux to solve buffer overflow problem using two policies - bounds-check recognition (BR) and pointer injection (PI). The approaches differ in tag propagation rules, the conditions that indicate an attack, and whether tagged input can ever be validated by application code. G. Edward Suh, Jaewook Lee, Srinivas Devadas [15] proved in their paper that DIFT can be used for Secure code execution.

6. MORE TOPICS

Intrusion detection is a major research area in security. It is the process of monitoring computer and network activities for any suspicious action and analyzing the activities to observe and discover any signs of intrusion in the system. The IDS or intrusion detection system implemented in software is the most common but is not absolute in security and is computationally slower. The recent research in hardware based intrusion detection and prevention system used along with software IDS has made it possible for faster processing and better security.

Software piracy is a big concern in the industry today. There is active research going in security domain to come up with effective means to identify software piracy and develop copy and tamper resistant software. The future of software is hosting the software on a hardware which is tamper proof and using the software directly from the hardware. One implementation of such a system is that the software is written on a hardware which is a form of execute-only memory (XOM). This allows for the software instructions to be executed directly from the dongle but these instructions cannot be altered or modified in any way. The software or instructions when copied from the hardware are encrypted and non-readable on any system. This ensures that the software would not execute from anywhere outside the given hardware.

7. CONCLUSION

With the growing amount of information and the ever-posing threat to this information, it is high time to take security as a prominent factor when designing applications or systems. 2014 has been a year of massive hacks. The critical information lost to Home Depot, Target and USPS exploits are a big security concern for all the users throughout the world. Software is known to have bugs and thus prone to exploits which can take control of the system. Hardware implementation, on the other hand, can be costly if it caters to a specific problem and cannot be used to solve others. It is difficult to achieve absolute security but a combination of software and hardware security features can take us a long way.

8. REFERENCES

[1] Information Security: Principles and Practice by Dr. Mark Stamp
[2] http://en.wikipedia.org/wiki/Computer_security
[3] Buffer Overflow:
http://en.wikipedia.org/wiki/Buffer_overflow
[3b] Buffer Overflow Protection:
http://en.wikipedia.org/wiki/Buffer_overflow_protection
[4] Address Space Layout Randomization:
http://en.wikipedia.org/wiki/Address_space_layout_randomization
[5] Hardware and Binary Modification Support for Code Pointer Protection From Buffer Overflow
Authors: Nathan Tuck, Brad Calder, George Varghese

[6] David Lie, Chandramohan Thekkath, and Mark Horowitz. Implementing an untrusted operating system on trusted hardware. In Symposium on Operating Systems Principles, October 2003
[7] Secure Bit2: Transparent, Hardware Buffer-Overflow Protection
http://www.cse.msu.edu/~cse825/SBit2.pdf
[8] http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/malware-reduction-general-technology.html
[9] Building a High-Performance, Programmable Secure Coprocessor
Authors: Sean W. Smith, Steve Weingart

[10] Secure mobile business information processing

Authors: Nicolai Kuntze, Roland Rieke, Karsten Sohr, Tanveer Mustafa, Kai-Oliver Detken

[11] CryptoManiac: A Fast Flexible Architecture for Secure Communication

http://web.eecs.umich.edu/~taustin/papers/ISCA01-cryptomaniac.pdf

[12] http://www.safenet-inc.com/data-encryption/hardware-security-modules-hsms/#sthash.5sjcarMj.dpuf

[13] http://www.maximintegrated.com/en/app-notes/index.mvp/id/5421

[14] Real-World Buffer Overflow Protection for Userspace & Kernelspace

Authors: Michael Dalton, Hari Kannan, Christos Kozyrakis

https://www.usenix.org/legacy/event/sec08/tech/full_papers/dalton/dalton_html/

[15] Secure Program Execution via Dynamic Information Flow Tracking

Authors: G. Edward Suh, Jaewook Lee, Srinivas Devadas