UNIVERSITY OF PIRAEUS

DIGITAL SYSTEMS DEPARTMENT

M.SC SECURITY OF DIGITAL SYSTEMS

MASTER THESIS

“Comparative analysis of the Windows Security XP, Vista,7, 8 and 10”

Gartaganis Charalampos   MTE 1504

Piraeus, Greece, February 2017

SUPERVISOR

Prof. Christoforos Dadogian

University Of Piraeus

Abstract

Over the past several years, Microsoft has implemented a number of memory protection and hard disk encryption mechanisms with the goal of preventing the exploitation of common software vulnerabilities on the Windows platform. Most of us own an appliance with software Windows, but do they have the required security? Nowadays it is difficult to say something is safe. Protection mechanisms such as GS, SafeSEH, DEP and ASLR complicate the exploitation of many memory corruption vulnerabilities. The purpose of this thesis is to determine the evolution of security protections of Windows and to build security awareness amongst users. This paper is divided into three parts. The first part provides an introduction on the subject of the thesis. In the second part we will analyze all versions of Windows separately and then we will process and present the changes of security protections in hard disk encryption and memory protection. In the third part we will attempt to collect all security protections, to facilitate as many users as possible. This thesis will review, analyze, conduct experiments and recommend several potential methods to mitigate the security breaches. Furthermore, this thesis hopes to offer the opportunity to figure out the difference between the security protections on Windows and the security protections added over the years.

Acknowledgements

I would first like to thank my professors for the wonderful experience I had at University of Piraeus. They taught me how to overcome the challenging and exciting problems of the real world and showed me how to use and utilize new technologies. They guided me and encouraged me when I felt lost in the world of digital security and taught me how to focus on detail. Moreover, I would like to thank my parents and my friends for their support during my studies. They were beside me, every single time I needed them and they were willing to help me and calm me down when I felt stressed. Without them I would not have found the motivation and energy needed to fulfill my dream.

List of Acronyms and Abbreviations

API	Application Programming Interface
ASLR	Address Space Layout Randomization
CVE	Common Vulnerabilities and Exposures
DEP	Data Execution Prevention
OS	Operating System
PAE	Physical Address Extension
PE	Portable Executable
PEB	Process Environment Block
ROP	Return Oriented Programming
SEH	Structured Exception Handling
SEHOP	Structured Exception Handling Override Protection

Contents

Contents of Figures

Contents of Tables

Prologue

The huge evolution of the Internet, has lead to daily conversion of the natural world into digital data electronic format. As almost all agencies, institutions, companies and individuals use computers with internet access, in most cases to manage their data, the value of the information gathered on the web gains enormous importance and is a subject that is increasingly discussed. Our dependence on these systems as well as the fact that their functionality and ease have grown significantly, has lead to their increased complexity.

As a result of this complexity we must deal with a multitude of weaknesses and problems in the security of systems and data, either by bugs or miss-configuration etc. The safety of users of communication networks has now become an important and highly interesting piece of modern technology. The universality of the Web and its widespread use in every-day life, from businesses to individuals, has stressed the importance of safe navigation and security protection. Inadequate security exposes users to harmful risks such as spam, viruses and identity theft data. But the security of networks and information handled is also a fairly complex issue. Different user categories may require different handling, for some it can as simple as incognito web browsing, a secure execution of financial transactions, or the proper functioning of a website, an entire company's network and their employees' private files.

Over the past several years, Microsoft has developed security protections for each of their editions such as implementing a number of hard disk encryption and memory protection mechanisms with the goal of preventing the exploitation of common software vulnerabilities on the Windows platform. Protection mechanisms such as GS, SafeSEH, DEP and ASLR complicate the many possible memory corruption insecurities and at first sight present an insurmountable obstacle for exploit attackers. Unfortunately, there are still several security issues on Windows, that may affect their users. As a result, Microsoft tries to upgrade the security protections with each edition and it seems to have been effective.

Chapter 1- Introduction

1.1 The most common Windows security vulnerabilities

1. File and share permissions that give up everything to everyone -This is easily the biggest vulnerability in Windows systems regardless of the type of system or Windows version. Users who create shares to make their local files available across the network are typically the culprits. Sometimes it's careless admins, other times they're honest mistakes. Unfortunately, all too often the "Everyone group" is given full access to every file on the system. Then, all it takes is for an insider to search for sensitive keywords stored in .pdf, .doc and other file formats. By using a text search tool such as Effective File Search the attacker will come across sensitive information that they shouldn't have access to.

2. Lack of personal firewall protection - This is another basic security control that's still not enabled on many Windows systems. Even the basic Windows Firewall can prevent connections to the IPC and ADMIN, shares that are often open and provide information and access that they shouldn't be divulging. Personal firewalls can also block malware infiltrations, wireless intrusions and more.

3. Weak or nonexistent drive encryption - The majority of organizations (large and small) do not use encryption. If a laptop or desktop computer is lost or stolen, the only way to prevent someone from cracking the Windows password and gaining full access to the hard drive is to encrypt everything using reasonable passphrases. Relying on Windows Encrypted File System (EFS) or other file/directory/volume-level encryption puts too much security control in the hands of users and is a breach waiting to happen.

4. No minimum-security standards - Users with wireless networks, especially, need to follow secure company policies at their homes, like requiring SSL for Outlook Web Access, a PPTP VPN connection for remote network connectivity or WPA-PSK with a strong passphrase to help ensure everything is safe and sound. This can be tough to enforce without a workstation-based wireless IDS/IPS (typically a component of an enterprise wireless management system) or a well-configured Network Access Control (NAC) system. Nevertheless, make it your policy and enforce it wherever possible.

5. Weak Windows security policy settings - Some examples of this include audit logging that is not being enabled for failed events and no password-protected screensavers. Polices to control these issues are easy to implement locally on each Windows system for smaller Windows shops not running Active Directory.

6. Unaccounted for systems running unknown and unmanaged services such as SQL Server Express
These are often legacy Windows systems that aren't within the scope of enterprise security and compliance. Sometimes, they're not even supported by third-party security management apps so they get pushed aside. These systems (typically Windows 98, NT and 2000) are often unhardened and unpatched and are waiting to be exploited. Inevitably there's going to be some random training or test system that everyone forgot about. But such a system is all it takes for someone with ill intent to get onto your network and do bad things.

Table 1: Attacks Statistics

$3 TRILLION Impact of lost productivity and growth	**$3.5 MILLION** **Average cost of a data breach (15% increase)**	**200+DAYS Median number of days attackers are present on a victim's network before detection**
46% of compromised systems had no malware on them	**23% of recipients opened phishing messages**	**50% of those who open, click attachments within the first hour**

1.2 Memory Protections Mechanisms

Among the main tasks of any OS is managing memory used by the OS and its processes. Memory management is the organizing of memory physically and logically, as well as memory sharing, protecting and relocating. A key property of memory management is memory virtualization. Virtual memory is an abstraction of the physical memory. Each process has its own virtual memory view. There are several reasons for the virtualization of memory. Amongst the most significant are the following:

- The amount of physical memory (RAM) and size of disk swapping vary in each machine setup. Memory virtualization enables a uniform memory layout, size and view for all processes. This enables the OS to handle memory regardless of hardware setup.
- Individual processes can operate with individual/isolated memory ranges or shared memory ranges.
- The abstraction layer introduces a platform for extended functionality in memory management, such as protection and optimization in a way that is suitable to a specific OS.

Memory management is handled by a memory management unit (MMU). This is a hardware component which interacts with the operation of the CPU. Several different operating modes exist for different computer architectures. The focus of this thesis is the IA-32 architecture in an operating mode as used by Microsoft Windows. The concepts of virtual memory and its management provide an opportunity to enforce access restrictions and protection. Segmentation and paging are the two types of memory organizing.

A segment is a memory range with a set of permissions and a given size. The CPU provides segment registers such as code-segment (CS), data-segment (DS) and stack-segment (SS). The use of segmentation has in most purposes been superseded by paging. In paging, paged virtual memory is divided into equally sized pages. Access restrictions can include a page being marked as accessible only to the kernel, or as read-only. A process cannot access a physical page that has not been mapped in its own page tables.

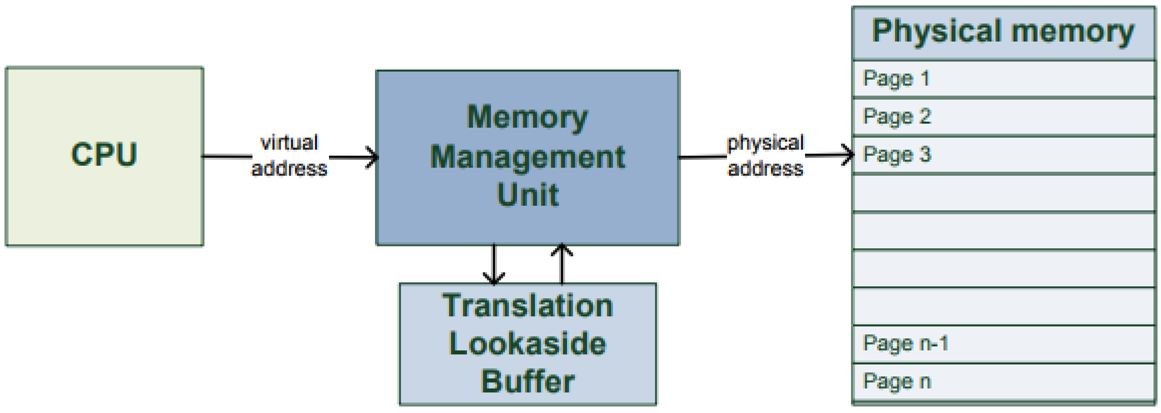


Figure 1

1.2.1 Limitations of paged memory protection

Three limitations in memory protection by the MMU has been identified. First, the fixed size of each page (commonly 4 kilobytes) is usually not representative of the size of the items contained in the page. A page may contain a set of different variables, objects and structures, with varying sizes. These may each have individual protection requirements. Due to collocation with other items with differing protection requirements, the lowest common protection level is used for any given page. The discrepancy between the protection on a per-page level and per-individual item in memory on a byte-level is referred to as the "protection granularity gap". Optimal page utilization has been chosen at the expense of protection granularity. Secondly, the access mode protection flag of a page can be possible to bypass.

One may assume a page marked as read-only is protected against writes. This, is not quite the case. A malicious kernel module may bypass the protection leveraged by the page flags by modifying the control register CR0. This register contains system control flags which control the operating mode and states of the CPU. One of the flags is named WriteProtect2 and is located at bit 16. If this flag is unset and the CPU is in kernel mode, the CPU is allowed to write to any page even if the page is read-only. A third technique to bypass memory protection is to locate the physical address of a protected page. It is then possible to create a new virtual memory mapping pointing to the same page, but with different protection flags.

Steps for buffer overflow when exploited:

- **Finding Input Shellcode Address:** When we send input data to the application, it should be such as to cause a buffer overflow to be triggered. But we must write the input data in such a way that we'll include the address that points to our shellcode that was also entered as part of the input data. This is needed so the program execution can later jump to the shellcode by using that shellcode memory address pointer.

- **Data Structure Overwrite:** In order for the program to actually jump to our shellcode, we must overwrite some data structure where the address is read from and jumped to. An example of such a structure is a function frame that's stored on the stack every time a function is called; besides all the other stuff stored on the stack, one of them is also the return EIP address where the function will return when it's done with the execution. If we overwrite that address with the address of our shellcode in memory, the program execution will effectively jump to that address and execute our shellcode.

- **Finding the System Function's Address:** When our shellcode is being executed, it will often call various system functions, which we don't know the addresses of. We can hardcode the addresses of the functions to call in the shellcode itself, but that wouldn't be very portable and it often won't work on modern systems because of the various memory protection mechanisms in place. In order to find the addresses of the functions that we want to call, we must dynamically find them upon the shellcode execution.

It is important to mention that if only one of the above conditions is not true, then the buffer exploitation will fail and our shellcode will be rendered non-executable. This is an essential observation because, if we're not able to satisfy every condition, the exploit will not work and the attack will be prevented.

To recap if we are able to prevent hackers from satisfying at least one of the above conditions, then the exploitation of the buffer overflow will fail. This is why we'll take a closer look at the methods for preventing each of these conditions.

Let's not forget about the instances where the buffer is very small and we can't actually exploit the condition in one go, but we must send multiple packets to the target in order to successfully exploit it; none such technique is the egg-hunting exploitation technique, where we first inject some malicious code into the process's memory, which contains an identifiable group of bytes often referred to as an egg. After, that we need to inject a second input into the process, which actually exploits and overwrites the return EIP address.

Thus, we can jump to the first part of the shellcode, which must find the egg in memory and continue the execution from there – this is the shellcode we've inputted into the process' memory with the first packet. At this point we should also mention that none of the exploitation techniques would be possible if programmers could write safe code sans bugs. Because we don't live in a ideal world, bugs exist and hackers are able to exploit these insecurities every day. In response, white-hat hackers have come up with various ways to prevent the exploitation of bugs even if they exist in the code: some of the techniques can be easily bypassed but others are very effective in protecting the process’ memory.

1.2.2 Techniques for Protecting the Finding of Input Shellcode Address

When exploiting a service, we’re sending input data to said service, which stores it in a buffer. The input data usually contains bytes that represent native CPU instructions. The buffer where our input bytes are written to can be located anywhere in memory:

- on the stack
- on the heap
- in the static data area

Currently, the following methods that prevent us from guessing the right input address, which we can use to overwrite some important data structure that enables us to jump to our shellcode, are:

- Instruction set randomization
- Randomizing parts of the operating system – ASLR

We need to overwrite the return address EIP or some other similar structure to jump to some instructions in memory that can help us eventually jump to our shellcode. This is why we must overwrite the structure with something meaningful, like an address that can take us to our shellcode and execute it. There are possibly infinite possibilities of how we can do that, but here's a list of a few of them to give us an idea of how this can be done:

- **call/jmp reg**: Here we’re using the register that contains the address of our shellcode, which we’re calling to effectively execute that shellcode. We can find either the “**call reg”** or **“jmp reg”** in one of the libraries the program needs to execute in order to jump to the shellcode. Note that this only works if one of the registers contains an address that points somewhere into the shellcode.
- **pop ret**: If we cannot use the previous option because none of the registers point somewhere in our shellcode, but there’s an address that points to the shellcode written on the stack somewhere, we can use multiple **pop** and one **ret** instructions to jump to that address.
- **push ret:** In the event that one of the registers points somewhere in our shellcode, we can also use the “push ret” instruction. This is particularly good if we cannot use the “call/jmp reg,” because we’re unable to find the appropriate jmp/call instructions in the loaded libraries. To solve that, we can push the address on the stack and then return to it by using the **push reg** and **ret** instructions.
- **jmp [reg + offset]:** We can use this instruction if there’s a register that points to our shellcode in memory, but doesn’t point to the beginning of our shellcode. We can try to find the instruction “jmp [reg+offset],” which will add the required bytes to the register and then jump to that address, presumably to the beginning of our shellcode.
- **Popad**: The “popad” instruction will pop double words from the stack into the general-purpose registers EDI, ESI, EBP, EBX, EDX, ECX and EAX. Also, the ESP register will be incremented by 32. By using this technique, we can make the ESP register point directly to our shellcode and then issue the **jmp esp** instruction to jump to it. If the shellcode is not present at the 32-byte boundary, we can add a number of nop instructions at the beginning of our shellcode to make it so.

- **forward short jump (jmp num):** We can use forward short jump if we want to jump over a couple of bytes. The opcode for the jmp instruction is 0xeb, so if we would like to jump over 16 bytes, we could use the **0xeb,0x10** instructions. We could also use conditional jump instructions, in which just the opcode is changed.
- **backward short jump (jmp num):** This is the same as forward short jump, except that we would like to jump backward. We can do this by using a negative offset number, where the 8-bit needs to be 1. If we would like to jump 16 bytes back, we could use the following instructions 0xeb,0x90.
- **SHE**: Windows applications have something called a default exception handler, which is provided by the operating system. Even if the application doesn't use exception handling, we can try to overwrite the SEH handler and then generate some exception, which will call that exception handler. Since we've overwritten the exception handler with the pointer to our shellcode, that will be executed when an exception occurs.

1.2.3 Techniques for Protecting Data Structure Overwrite

One of the conditions already mentioned was that if we can overwrite certain data structure, we might be able to gain control of the program and possibly the whole system. The data structures that we can overwrite are the following:

- **EIP in Stack Frame**: When a function is called, it is stored in a function frame on the stack, which also contains the EIP that points to the next instruction after the function call, which is required so that the function knows where to return to. If we overwrite the EIP return address, we'll be able to jump and execute the instruction on the arbitrary memory address.
- **Function Pointers**: We can also overwrite a function pointer that points to a function on the heap. If we overwrite the function pointer to point to our shellcode somewhere in memory, that shellcode will be executed whenever the function is called (the one whose pointer address we've overwritten). Keep in mind that function pointers can be allocated anywhere in memory, on stack, heap, data, etc.
- **Longjmp Buffers**: The C programming language has something called a checkpoint/rollback system called setjmp/longjmp. The basic idea is to use the setjmp (buffer) to go to the checkpoint and then use the longjmp (buffer) to go back to the checkpoint. But if we manage to overflow the buffer, the longjmp (buffer) might jump back to our shellcode instead.
- **SEH Overwrite**: We can overwrite stack exception handling structures stored on the stack, which can allow us to gain code execution when an exception occurs.
- **Program Logic**: With this kind of attack, we're overflowing the arbitrary data structure that a program uses incorrectly: by possibly executing the inputted shellcode. This is an extremely rare occurrence of a bug, but it is possible.

We can use the following techniques to defend the data structures:

- **Non-Executable Stack**: If the stack is non-executable, then we won't be able to execute the code written to the stack. This protection is very effective against stack buffer overflow attacks, but doesn't protect us from overflowing other data structures and executing the shellcode from there.
- **DEP (Data Execution Prevention):** When DEP is enabled, we can't execute the code from pages that are marked as data. This is an important observation, because with typical problems the code is not contained and executed from stack or heap structures, which typically contain only data.
- **Array Bounds Checking**: With this prevention mechanism, the buffer overflows are completely eliminated, because we can't actually overflow an array, since the bounds are being checked

automatically (without user intervention). However, this approach rather slows down the program execution, since every access to an array must be checked to see if it’s within the bounds.

- **Code Pointer Integrity Check**: This protection provides a way to check whether the pointer has been corrupted before trying to execute the data from that address.

DEP (Data Execution Prevention)

DEP allows the pages in memory to be marked as non-executable, which prevents the instructions in those pages to be executed (usually the stack and heap data structure are used with DEP enabled). If the program wants to execute code from the DEP-enabled memory page, an exception is generated, which usually means the program will terminate. The default setting in a Windows system is that DEP is enabled. To check whether DEP is enabled, we must look into the C:\boot.ini configuration file and look at the **/noexecute** flag value. The values of the /noexecute switch can be one of the following values:

- OptIn: DEP enabled for system modules, but user applications can be configured to also support DEP.
- OptOut: DEP enabled for all modules and user applications, but can be disabled for certain user applications.
- AlwaysOn: DEP enabled for all modules and all user applications and we can’t switch it off for any user application.
- AlwaysOff: DEP disabled for all modules and all applications and we can’t switch it on for any user application or any system module.

The reason for this is that hackers could use a hardcoded address in their exploits, which would work on various versions of Windows systems, because the virtual/linear addresses of the programs are always the same. Windows Vista presented a concept known as address space layout randomization or ASLR that loads the executables and libraries that support ASLR at arbitrary base address every time the system reboots. This makes it difficult to hardcode the addresses in the exploits, but it is still possible. This is due to the fact that the executable and all the libraries the process uses must be compiled with ASLR enabled, but this is often not the case.

The developers almost never compile the executable with ASLR enabled. Furthermore some system libraries provided by Microsoft are not always compiled with ASLR enabled. The attacker needs only one address to be at a constant place in order to exploit a buffer overflow thus he only needs one executable or one loaded library file to not be compiled with ASLR enabled. As it’s still often the case that at least some library doesn’t support ASLR, ASLR protection is rather easy to bypass and thus not really effective. Windows XP doesn’t support ASLR, which is one reason among many to switch to a newer system, such as Windows Vista or Windows 7. ASLR doesn’t affect just the base address where the libraries are loaded, but also the base address of stack, heap and other stuff that must be loaded into memory. But the DLL libraries are loaded at the same address regardless of the process using them; this is set up during the boot process of Windows.

1.3 Exploit Mitigations in Windows

Newer software releases, like Windows 7 and the 2007 Microsoft Office system, are consistently less prone to active exploitation than older releases. These are some of the significant exploit mitigation technologies that have been added to Windows over the past few years. Windows incorporates a number of defensive strategies to protect customers from attack. Some of these defenses are part of the core operating system and additional defenses are offered by the Microsoft Visual C++ compiler. The defenses include:

- **Data Execution Prevention (DEP):** Buffer overflow attacks, in which an attacker forces a program or component to store malicious code in an area of memory not intended for it, are some of the most common exploits seen today. DEP is a Windows feature that enables the system to mark one or more pages of memory as non-executable. Marking memory regions as non-executable means that code cannot be run from that region of memory, which makes it harder for exploits involving buffer overruns to succeed. DEP was introduced in Windows XP SP2 and has been included in all subsequent releases of Windows desktop and server operating systems. For application compatibility reasons, DEP is “opt-in” in Windows XP, Windows Vista, and Windows 7.

- **Address Space Layout Randomization (ASLR):** In older versions of Windows core processes tended to be loaded into predictable memory locations upon system startup. Some exploits work by targeting memory locations known to be associated with particular processes. ASLR randomizes the memory locations used by system files and other programs, making it much harder for an attacker to correctly guess the location of a given process. The combination of ASLR and DEP creates a fairly formidable barrier for attackers to overcome in order to achieve reliable code execution when exploiting vulnerabilities. ASLR was introduced in Windows Vista and has been included in all subsequent releases of Windows.

- **SafeSEH:** The SafeSEH protection mechanism is designed to prevent attackers from taking control of the program execution by overwriting an exception handler record on the stack. If a binary is linked with the /SafeSEH linker option, its header will contain a table of all valid exception handlers within that module. When an exception occurs, the exception dispatcher code in NTDLL.DLL verifies that the exception handler records on the stack points to one of the valid handlers in the table. If the attacker overwrites the exception handler record and points it somewhere else, the exception dispatcher will detect this and terminate the program.When an exception is raised at run time, the operating system (Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003, or Windows XP with Service Pack 2 or 3) will not dispatch to an address in that image unless it is one of the exception handler addresses in the PE header.

- **GS (compiler flags)**: Introduced in Visual Studio .NET 2002 and used by all supported IE versions, it is a compiler technology which adds a Buffer Overrun Detection capability to an application’s stack. Security secrets known as “canaries” are added to application’s stack boundaries at runtime. For example, a canary placed between a stack buffer and a return address, will be overwritten by a buffer overflow attack targeting the return address. This provides a means for the process to detect that an overflow has occurred. An exception will be raised and the process can be safely terminated before the attacker’s code is run.

- **Structured Exception Handler Overwrite Protection (SEHOP):** Another common technique used by exploit writers is to overwrite an exception handler to gain code execution. SEHOP stops this entire class of exploits from working by verifying that a thread’s exception handler list is intact before allowing any of the registered exception handlers to be called. SEHOP was introduced in Windows Vista SP1 and Windows Server 2008, Microsoft introduced support for Structured Exception Handler Overwrite Protection.

1.3.1 What’s the difference between SafeSEH and SEHOP?

On the surface, SAFESEH and SEHOP, appear the same: both help mitigate the severity of attacks that attempt to overwrite exception handlers. SEHOP is a more complete defense because it can protect against handlers that point to non-SafeSEH modules and handlers that point to a non-image region, such as the heap. Either SafeSEH or SEHOP can be used, but the latter is preferred. The best solution is to build your code to use both: SafeSEH will work on versions of Windows prior to Windows Vista SP1, and SEHOP provides a more comprehensive defense on Windows Vista SP1 and later.

Memory protection is based on the division of the memory of a digital computer, either by software or by hardware, into a sequence of segments and the providing of each segment or group of segments with a code key, which is stored in the same memory or in a special memory. During a memory cycle the segment and the corresponding key are determined; then the key is compared with the authorized key of the memory protection, which is specified in the same instruction or in a supervisory program. A discrepancy in the keys is regarded as a memory protection violation, and the execution of the program is interrupted. The interruption of the program is organized so that the protected region of memory is left unchanged. Memory protection functions during every memory cycle, either in the cycle for writing information or in the information reading cycle, or in both.

Memory protection preserves the contents of specified regions of memory from loss of information during program execution as a result of erroneous transmission of information caused by failures or breakdowns in the equipment or in the supervisory program of the digital computer, or those caused by mistakes of the programmer or user; it also prevents information from falling into the hands of an unauthorized user through accidental or intentional interference. Hardware methods of memory protection include storage devices for the protection keys; the capacity of these devices corresponds to the number of protected segments, and the speed is greater by a factor of 10 than that of the digital computer's working memory.

In addition, there are circuits for the comparison of the protection keys and for the interruption and indication of memory protection violations. Software means of memory protection include programs for monitoring the memory segments, encoding them, and compiling tables of correspondence; programs for the dynamic redistribution of memory protection according to user's instructions and according to the parameters of the simultaneously solved tasks; and programs for the analysis of the causes of memory protection violations and the adoption of solutions for their elimination.

Memory protection increases the efficiency of digital computer operation, reducing time losses caused by the search for errors and repeated computation resulting from loss of information. Memory protection is necessary for the simultaneous solution of several problems on one digital computer in a time-sharing system, for the simultaneous service of several users, for the use of program libraries and archives belonging to several users, and for the simultaneous operation of several on-line units.

Disk encryption is the encryption of an entire disk not just specific files. In other words, if you open up your computer and remove the hard drive, all the contents of that physical hard drive are encrypted. Disk encryption is also known as hard drive encryption, full disk encryption, whole disk encryption, and partial combinations of these three (hard disk encryption, full hard disk encryption). If anyone or anything alludes to an entire disk being encrypted, chances are this is what they're talking about. The real-world counterpart to disk encryption is the use of a safe with a built-in lock. That is, if you place any documents and close the door of the safe, the documents are protected. The only way to get back those documents is by knowing the combination or having the key to the lock, or busting the safe's door open. Likewise, any files that you save on a computer or digital device with full disk encryption will be encrypted automatically due to the fact that disk encryption is being used.

Chapter 2: Version of Windows

2.1 Windows XP

This operating system has stood the test of time. But while Windows XP may be Microsoft's most popular OS, when it comes to security, it is sorely lacking.

- The most widely targeted system in mass exploitation for botnets and key-loggers.
- Attack surface reduction has reduced the number of vulnerabilities in services, but client software is almost completely unprotected.
- Reliable exploitation techniques exist for almost all types of vulnerabilities.
- Many Windows XP machines have turned into breeding grounds for malware.

Real-Time Malware Protection: Windows Defender provides real-time protection against malware and potentially unwanted software out of the box.

BitLocker Drive Encryption: BitLocker Drive Encryption enables users and administrators to encrypt entire hard drives, protecting data on lost or stolen computers from unauthorized access. BitLocker is a full disk encryption feature. It is designed to protect data by providing encryption for entire volume. By default, it uses the AES encryption algorithm in cipher block chaining (CBC) with a 128-bit or 256-bit key.

There are three authentication mechanisms that can be used as building blocks to implement BitLocker encryption:

- **Transparent operation mode**: This mode uses the capabilities of TPM 1.2 hardware to provide for a transparent user experience every time the user powers up and logs into Windows as normal. The key used for disk encryption is encrypted by the TPM chip and will only be released to the OS loader code if the early boot files appear to be unmodified. The pre-OS components of BitLocker achieve this by implementing a Static Root of Trust Measurement.
- **User authentication mode**: This mode requires that the user provide some authentication to the pre-boot environment in the form of a pre-boot PIN or password.
- **USB Key Mode**: The user must insert a USB device that contains a startup key into the computer to be able to boot the protected OS. Note that this mode requires that the BIOS on the protected machine supports the reading of USB devices in the pre-OS environment. The key may also be provided by a CCID for reading a cryptographic smartcard. Using CCID provides additional benefits beyond just storing the key file on an external USB thumb drive, because the CCID protocol hides the private key using a cryptographic processor embedded in the smartcard; this prevents the key from being stolen by simply being read off the media on which it is stored.

DEP (Data Execution Prevention)

Starting from WinXP SP2 and Win Server 2003 SP1, Windows has implemented a new security feature to prevent code execution from non-executable memory ranges. DEP (Data Execution Prevention) comes in two flavors.

Hardware Enforced DEP: The CPU marks pages of memory as non-executable

Software Enforced DEP: Alternative for CPU's that do not support these features.

CPU's that support hardware enforced DEP will refuse to execute code from memory ranges that have the non-executable (NX) bit set. The main reason for this is to prevent custom/malicious code from being injected into another program and to then be executed. This was mainly implemented to put up hurdles for malware and stack-based exploits. However DEP can sometimes cause programs to behave in unintended and erroneous ways because it prevents legitimate processes from doing things they are supposed to do. To solve this problem DEP can be configured in two ways on your host operating system.

Opt-In Mode: DEP is only enabled for system processes and specifically defined programs. **Opt-Out Mode**: DEP is enabled for all programs and services except those specifically/manually disabled.

2.2 Windows Vista

Since its initial creation, there have been two service packs and dozens of individual security patches for Vista. But despite its many vulnerabilities and the fact that it may be the most despised OS, Vista has always been more secure than Windows XP. Features such as address-space randomization, data execution prevention, application isolation and User Access Control (UAC) protect Vista from many of the exploits that work so well against Windows XP.

- Limited deployment, not a target for mass exploitation.
- More attack surface reduction in services, but client software still an easy target.
- ASLR and DEP are very effective in theory, but backwards compatibility limitations severely weaken them.

2.2.1 Authentication

Windows Vista continues to have built-in authentication support for passwords and smart cards, which makes it simpler for developers to add their own custom authentication methods to Windows, such as biometrics and tokens. Also provides enhancements to the Kerberos authentication protocol and smart card logons. Deployment and management tools, such as self-service personal identification number (PIN) reset tools, make smart cards easier to manage.

Benefits: The smart card improvements in Windows Vista make it easier for organizations to deploy and support this built-in authentication method. Windows Vista directly benefits developers who offer customized authentication mechanisms such as biometrics and tokens by making it easier to implement the authentication mechanism.

2.2.2 Anti-Malware

User Account Control can reduce the impact of malware on Windows Vista. In addition, Windows Vista can clean many worms, viruses, rootkits and spyware, thereby ensuring the integrity of the operating system and the privacy of users' data. Windows Vista will also include Windows Defender, a technology that helps protect your computer against pop-ups, slow performance and security threats caused by spyware and other unwanted software. It features Real-Time Protection, a monitoring system that recommends actions against spyware when it's detected, and a new streamlined interface that minimizes interruptions and helps you stay productive.

Benefits: Malware often degrades system performance, which often leads users to prematurely conclude that their computers are too slow or unreliable and need to be re-imaged. Unfortunately, this process increases computer maintenance costs overall. Malware's greatest threat, however, is to security. For example, malware may compromise confidential data or introduce additional security vulnerabilities to a computer. Therefore, the added protection and malware cleaning available in Windows Vista improves the performance and security of the computers on your network.

2.2.3 Data Protection

Theft or loss of corporate intellectual property is an increasing concern for organizations. Windows Vista has improved support for data protection at the document, file, directory, and machine level. The integrated Rights Management client allows organizations to enforce policies around document usage. The Encrypting File System, which provides user-based file and directory encryption, has been enhanced to allow storage of encryption keys on smart cards, providing better protection of encryption keys. In addition, the new BitLocker Drive Encryption enterprise feature adds machine-level data protection.

On a computer with appropriate enabling hardware, BitLocker Drive Encryption provides full volume encryption of the system volume, including Windows system files and the hibernation file, which helps protect data from being compromised on a lost or stolen machine. In order to provide a solution that is easy to deploy and manage, a Trusted Platform Module (TPM) chip is used to store the keys that encrypt and decrypt sectors on the Windows hard drive. It requires the TPM and an enterprise management infrastructure to ensure that the feature is easy to use for end users.

A TPM chip is a hardware component available in some newer computers that stores keys, passwords, and digital certificates. BitLocker also stores measurements of core operating system files in a TPM chip. Every time the computer is started, Windows Vista verifies that the operating system files have not been modified in an offline attack. An offline attack is a scenario where an attacker boots an alternative operating system in order to gain control of the system. If the files have been modified, Windows Vista alerts the user and refuses to release the key required to access Windows. The system then goes into a recovery mode, prompting the user to provide a recovery key to allow access to the boot volume.

Benefits: Windows XP and earlier versions of Windows are vulnerable to offline attacks that attempt to obtain a user's data on lost or stolen computers. Unlike online attacks, which occur when the operating system is running, offline attacks occur when the operating system is turned off. The most common types of offline attacks are:

- Starting an offline computer with a boot disk and retesting the administrator password so that the attacker can start the operating system and authenticate.
- Accessing the computer's hard disk directly with a different operating system to bypass file permissions.

2.2.4 ASLR (Address Space Layout Randomization)

ASLR (Address Space Layout Randomization) is amongst the most popular software applications. ASLR was introduced in Windows Vista and combats attacks by loading program modules into a different, random area of memory each time, but it only works for applications that explicitly enable it when they are compiled. A different issue is ASLR support wasn’t built into Windows XP, which is still the dominant Windows platform. Data Execution Prevention (DEP), on the other hand, was introduced in Windows XP Service Pack 2 and was enabled in 71% of the applications Microsoft studied. Visual Studio 2010 enables ASLR by default. The technology is well documented and there are several Microsoft resources online that provide step-by-step instructions on how to incorporate it into an application. Microsoft’s SDL initiative for creating a systematic approach to secure software development has made Windows more secure.

This is one reason why we’re seeing more attacks against other applications rather than Windows itself. ASLR is important security feature enterprises should incorporate into their applications. Even though it may be something new for developers to learn, it's not difficult, and applications will be a lot more secure as a result. Software vendors who don’t support ASLR and other security initiatives in their applications are likely to be heavily targeted if those are adopted by their competitors.

If you are concerned applications running on your Windows XP machines are vulnerable to ASLR-based attacks, consider using Microsoft’s Enhanced Mitigation Experience Toolkit 2.0(EMET), which can retroactively apply various security mitigation technologies, including Mandatory Address Space Layout Randomization (Mandatory ASLR) to selected applications. Mandatory ASLR can force the operating system to load a dynamic link library written before ASLR was available to a random location regardless of the flags it was compiled with. Another advantage of using EMET is that you don’t need to recompile your own in-house software to set the various flags to enable DEP and ASL

2.3 Windows 7

Built on top of the Vista kernel, Windows 7 isn't all that different from its predecessor and when it comes to security, Windows 7 doesn't offer a lot more than Vista:

BitLocker to Go: which allows for the encryption of removable media
AppLocker: which provides greater control for regulating the applications that users access, are the only two new security features. But Microsoft did make changes to User Access Control (UAC) in Windows 7. UAC was a new feature in Vista that prompted the user for permission any time a change was going to be made to the system.

- Minor exploit mitigation changes since Vista.
- Potential for a wide deployment.
- Improved support for DEP and ASLR from Microsoft and third party vendors.

2.3.1 Bottom-up & top-down randomization

1) Heaps and stacks are randomized
2) PEBs/TEBs are randomized, but with limited entropy
3) VirtualAlloc and MapViewOfFile are not randomized
4) Predictable memory regions can exist as a result

2.3.2 AppLocker

Administrators can use Group Policy to keep users from running particular programs that might present a security threat. But they've never been used that much because they aren't easy to operate. Windows 7 has improved on the concept with a new feature called AppLocker. AppLocker is also included in Windows Server 2008 R2. It's easier to use and gives administrators more flexibility and control. You can use AppLocker with domain Group Policies or on the local machine with the Local Security Policy snap-in.

2.3.3 Protecting Data from Unauthorized Viewing

Windows 7 retains the data protection technologies available in Windows Vista like the Encrypting File System (EFS), built-in Active Directory Rights Management Services technology, and granular USB port controls. In addition to the incremental updates in these technologies, Windows 7 provides several significant improvements to the popular BitLocker Drive Encryption technology.

2.3.4 BitLocker and BitLocker to Go

Windows 7 addresses the continued threat of data leakage with manageability and deployment updates to BitLocker Drive Encryption and the introduction of BitLocker to Go: enhanced data protection against data theft and exposure by extending BitLocker support to removable storage devices. Whether traveling with your laptop, sharing large files with a trusted partner, or taking work home, BitLocker and BitLocker to Go protected devices help ensure that only authorized users can read the data, even if the media is lost, stolen, or misused. Best of all, BitLocker protection is easy to deploy and intuitive for the end user, all the while leading to improved compliance and data security. BitLocker to Go also gives administrators control over how removable storage devices can be utilized within their environment and the strength of protection that they require. Policies are also available to require appropriate passwords, smart card, or domain user credentials to utilize a protected removable storage device.

2.4 Windows 8

In Windows 8, Windows Defender replaces Microsoft Security Essentials. Windows Defender runs in the background and notifies you when you need to take specific action. However, you can use it anytime to scan for malware if your computer isn't working properly or you clicked a suspicious link online or in an email message.

Bottom-up & top-down randomization:

- All bottom-up/top-down alloca-Yons are randomized
- Accomplished by biasing start address of alloca-Yons
- PEBs/TEBs now receive much more entropy
- Both are opt-in (EXE must be dynamic base)

2.4.1 AppContainers and Vulnerability Mitigations

There are two primary reasons driving these impressive results on mobile devices. The first being the fact that all apps come from a centralized store that vets the apps before making them available to customers. Second, all of these apps run inside of a sandbox called the AppContainer. The AppContainer utilizes a sandboxing technology which is effective at preventing malicious apps from tampering with the system, other apps, and your data. Windows 8.1 also utilizes this technology making the system less susceptible to attacks even in the event that vulnerabilities are discovered. Improvements to technologies like ASLR and DEP where made in Windows 8.1 to ward off attackers and close said vulnerabilities.

2.4.2 Address space layout randomization

One of the most common techniques used to gain access to the system is to find a vulnerability in a privileged process that is already running, guess or find a location in memory where important system code and data have been placed, and then overwrite that information with a malicious payload. In the early days of operating systems, this could be done by any malware that could write directly to the system memory; the malware would simply overwrite system memory within well-known and predictable locations.

Address space layout randomization (ASLR) makes that type of attack much more difficult by randomizing how and where important data is stored in memory. With ASLR, it is more difficult for malware to find the specific location it needs to attack. ForceASLR is arguably the most important change to ASLR in Windows 8. ForceASLR is a new loader option used by Internet Explorer 10 to instruct the operating system to randomize the location of *all* modules loaded by the browser, even if a given module was not compiled with the /DYNAMICBASE flag. The ForceASLR protection was added to the Windows 8 kernel, and the feature is now available as an update to Windows 7 that will be installed when Internet Explorer 10 is installed on that platform. To help ensure compatibility with this feature, and to provide memory-randomization protection to older Internet Explorer versions that don't support ForceASLR, we continue to recommend that add-on developers make use of the /DYNAMICBASE flag.

Although the ASLR implementation in Windows 7 was effective, it wasn't applied holistically across the Windows system, and the level of entropy (cryptographic randomization) wasn't always at the highest possible level. To decrease the likelihood that sophisticated attacks such as heap spraying could succeed, Microsoft applied ASLR holistically across the system and increased the level of entropy many times. The ASLR implementation in Windows 8 is greatly improved over Windows 7 especially with 64-bit system and application processes that can take advantage of a vastly increased memory space, making it even more difficult for malware to predict where Windows 8 stores vital data. When used on systems with TPMs, ASLR memory randomization will be increasingly unique across devices, making it even more difficult for a successful exploit that works on one system to work reliably on another.

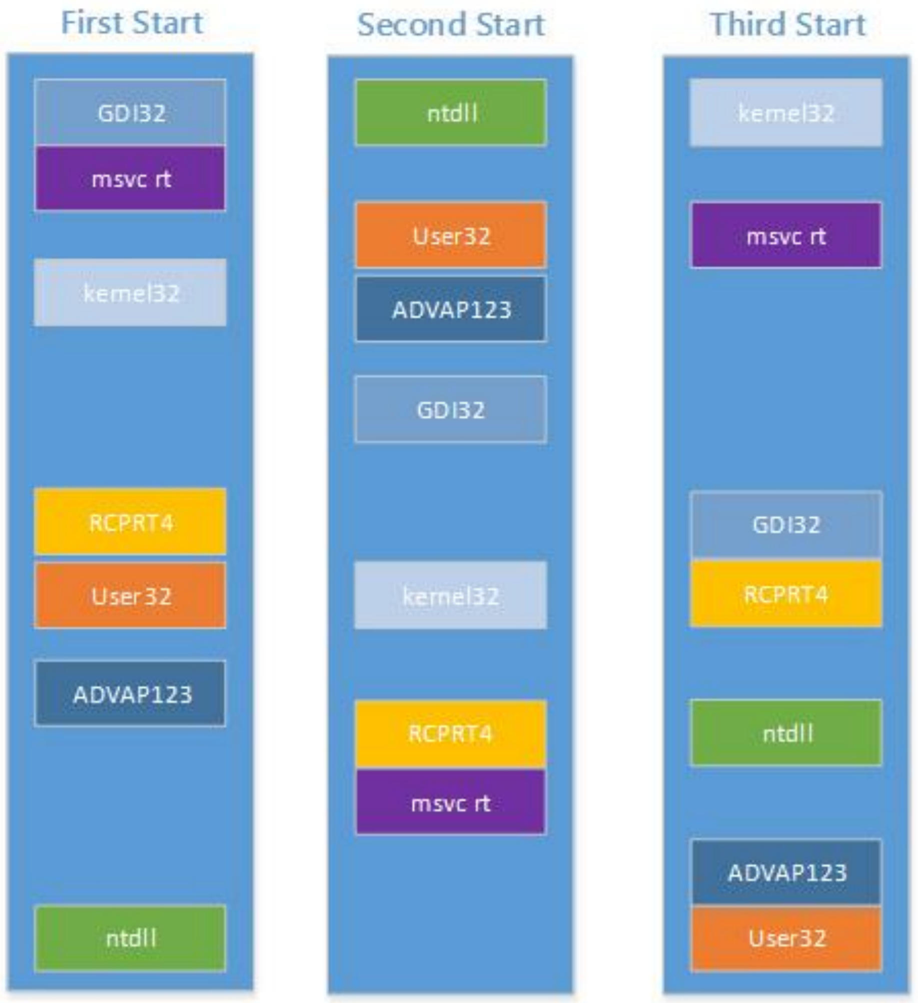


Figure 2

Windows 8 applies ASLR holistically across the system and increases the level of entropy many times compared with previous versions of Windows to combat sophisticated attacks such as heap spraying. 64-bit system and application processes can take advantage of a vastly increased memory space, which makes it even more difficult for malware to predict where Windows 10 stores vital data. When used on systems that have TPMs, ASLR memory randomization will be increasingly unique across devices, which makes it even more difficult for a successful exploit that works on one system to work reliably on another.

2.4.3 Data execution prevention

Malware depends on its ability to put a malicious payload into memory with the hope that it will be executed later, and ASLR is going to make that much more difficult. Data execution prevention (DEP) does exactly that and substantially reduces the range of memory that malicious code can use for its benefit. DEP uses the No execute (NX) bit on modern CPUs to mark blocks of memory as data that should never be executed as code. Therefore, even if an attacker succeeds in loading the malware code into memory, they will not be able to execute it. Because of the importance of DEP, Windows 8 is the first version of Windows that requires a processor that includes hardware-based DEP support. Users cannot install Windows 8 on a computer that does not have DEP enabled. Fortunately, most processors released since the mid-2000s support DEP.

2.4.4 Windows Heap

The heap is a location in memory that Windows uses to store dynamic application data. Windows 8 improves on the Windows 7 heap design by mitigating the risk of heap exploits that could be used as part of an attack to successfully compromise Windows 7.

Windows 8 has several important improvements to the heap, including:

1) Internal data structures used by the heap are now better protected against memory corruption.
2) Heap memory allocations now have randomized locations and sizes, making it harder for an attacker to predict the location of critical memory to overwrite. Specifically, Windows 8 adds a random offset to the address of a newly allocated heap, making the allocation much less predictable.
3)Windows 8 also adds "guard pages" before and after blocks of memory. If an attacker attempts to write past a block of memory (a common technique mentioned in previous chapters, known as a buffer overflow), the attacker will have to overwrite a guard page. Any attempt to modify a guard page is considered a memory corruption, and Windows 8 responds by instantly terminating the app. Windows 8 resolves known heap attacks that could be used to compromise a PC running Windows Vista or Windows 7.

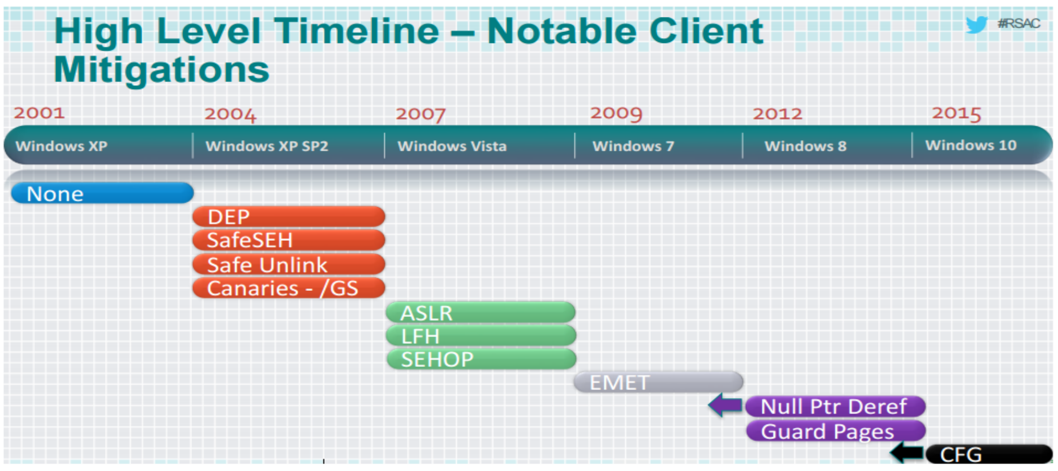


Figure 3

2.4.5 What is Control Flow Guard

Control Flow Guard (CFG) is a highly-optimized platform security feature that was created to combat memory corruption vulnerabilities. By placing tight restrictions on where an application can execute code from, it makes it much harder for exploits to execute arbitrary code through vulnerabilities such as buffer overflows. CFG extends previous exploit mitigation technologies such as /GS, DEP, and ASLR.

Table 2: Evolution of Windows

2001 – Windows XP - Security for mitigating binary exploitation on Windows is virtually non-existent at this point
2004 – Windows XP Service Pack 2 - DEP - SafeSEH, - GS Cookies - Stack & Heap marked non-executable
2006 – Windows Vista - ASLR is implemented, applies to stack/heap/images. - Hardened Heap - SEHOP
2009 – Windows 7 - DEP & ASLR further improved - Security wise, not too different from Vista, but widespread adoption.
2012/2013 – Windows 8/8.1 - Windows 8 came with a huge focus on beefing up security

Microsoft calls device encryption**.** While it has very specific hardware requirements, the feature is designed to improve local security for Windows users without them ever needing to know about it. Windows 8 is new device encryption treats your x86-based Windows tablet or laptop more like an ARM-based tablet or smartphone. Rather than requiring a user or system administrator to enable it, your device’s boot partition comes encrypted out of the box. This encryption is essentially invisible during normal use you pick up the tablet, log in, and use it just as you would an unencrypted PC. If someone were to steal the device from you, though, they wouldn’t be able to get at any of your information without your account password or your encryption key, which in this case is protected by your account password. When you first fire up Windows 8 on a PC that supports the feature, head to the “PC Info” section in the device settings screen to check your encryption status. Computers with the necessary hardware features begin encrypting the drive immediately, but the master key needed to decrypt the drive isn’t protected.

A user with administrator access will have to log in with a Microsoft account, at which point the device will generate a recovery key and upload it to Microsoft’s servers. This recovery key can then be accessed from another computer with your Microsoft account if you’re ever locked out of your system. Active Directory user accounts can also be used to store the key, provided your domain administrator has enabled the proper Group Policy settings. This is a far cry from the standard BitLocker encryption process, which requires individuals to back up and store their own key manually and must be enabled by users themselves. However, with the exception of the part where your key is uploaded to Microsoft’s servers, the underlying technology is exactly the same as it is in BitLocker. The nice thing about the automated device encryption is that it extends to every edition of Windows 8, where BitLocker is a Pro- or Enterprise-tier feature in Windows 8 and an Ultimate- and Enterprise-tier feature in Windows 7 or Vista.

2.5 Windows 10

Data Execution Prevention (DEP) helps prevent exploitation of buffer overruns. Also, **Data Execution Prevention (DEP)** is a system-level memory protection feature available in Windows operating systems. DEP enables the operating system to mark one or more pages of memory as non-executable, which prevents code from being run from that region of memory, to help prevent exploitation of buffer overruns. **DEP** helps prevent code from being run from data pages such as the default heap, stacks, and memory pools.

SEHOP helps prevent overwrites of the **Structured Exception Handler. Structured Exception Handling Overwrite Protection (SEHOP)** is designed to help block exploits that use the Structured Exception Handler (SEH) overwrite technique. Because this protection mechanism is provided at run-time, it helps to protect apps regardless of whether they have been compiled with the latest improvements. A few applications have compatibility problems with SEHOP, so be sure to test for your environment.

ASLR helps mitigate malware attacks based on expected memory locations. **Address Space Layout Randomization (ASLR)** loads DLLs into random memory addresses at boot time. This helps mitigate malware that's designed to attack specific memory locations, where specific DLLs are expected to be loaded.

2.5.1 Identity and access control

Microsoft Passport

Microsoft Passport provides strong two-factor authentication (2FA), fully integrated into Windows, and replaces passwords with the combination of an enrolled device and either a PIN or Windows Hello. Microsoft Passport is conceptually similar to smart cards but more flexible. Authentication is performed by using an asymmetric key pair instead of a string comparison (for example, password), and the user’s key material can be secured by using hardware. Unlike smart cards, Microsoft Passport does not require the extra infrastructure components required for smart card deployment. In particular, you do not need public key infrastructure (PKI). Microsoft Passport offers three significant advantages over the current state of Windows authentication: It’s more flexible, it’s based on industry standards, and it effectively mitigates risks.

Brute-force attack resistance

A brute-force attack is the process used to break into a device simply by guessing a user’s password, PIN, or even his or her biometric identity over and over until the attacker gets it right. Over the last several versions of Windows, Microsoft has added features that dramatically reduce the chances that such an attack would succeed. The Windows 7 operating system and previous versions defended against brute-force attacks in a straightforward way: they slowed or prevented additional guesses after multiple mistakes. When users use a full password to log on, Windows forces users to wait several seconds between attempts if they type their password incorrectly multiple times.

You can even choose to have Windows lock out an account for a period of time when it detects a brute-force attack. Windows 8.1 and Windows 10 support an even more powerful -but optional- form of brute-force protection when the credentials are tied to TPM. If the operating system detects a brute-force attack against the Windows sign-in and BitLocker protects the system drive, Windows can automatically restart the device and put it in BitLocker recovery mode until someone enters a recovery key password. This password is a virtually unguessable 48-character recovery code that must be used before Windows can be able to start normally.

2.5.2 Information Protection

Prepare for drive and file encryption

The best type of security measures are transparent to the user during implementation and use. Every time there is a possible delay or difficulty because of a security feature, there is strong likelihood that users will try to bypass security. This situation is especially true for data protection, and that's a scenario that organizations need to avoid. Whether you're planning to encrypt entire volumes, removable devices, or individual files, Windows 10 meets your needs by providing streamlined, usable solutions. In fact, you can take several steps in advance to prepare for data encryption and make the deployment quick and smooth.

2.5.3 Deploy hard drive encryption

BitLocker is capable of encrypting entire hard drives, including both system and data drives. BitLocker pre-provisioning can drastically reduce the time required to provision new PCs with BitLocker enabled. With Windows 10, administrators can turn on BitLocker and the TPM from within the Windows installation Environment before they install Windows or as part of an automated deployment task sequence without any user interaction. Combined with Used Disk Space Only encryption and a mostly empty drive (because Windows is not yet installed), it takes only a few seconds to enable BitLocker. With earlier versions of Windows, administrators had to enable BitLocker after Windows had been installed. Although this process could be automated, BitLocker would need to encrypt the entire drive, a process that could take anywhere from several hours to more than a day depending on drive size and performance, which significantly delayed deployment.

Device encryption

Beginning in Windows 8.1, Windows automatically enables BitLocker device encryption on devices that support InstantGo. With Windows 10, Microsoft offers device encryption support on a much broader range of devices, including those that are InstantGo. Microsoft expects that most devices in the future will pass the testing requirements, which make device encryption pervasive across modern Windows devices. Device encryption further protects the system by transparently implementing device-wide data encryption.

Unlike a standard BitLocker implementation, device encryption is enabled automatically so that the device is always protected. The following list outlines how this happens:

- When a clean installation of Windows 10 is completed and the out-of-box experience is finished, the computer is prepared for first use. As part of this preparation, device encryption is initialized on the operating system drive and fixed data drives on the computer with a clear key (this is the equivalent of standard BitLocker suspended state).
- If the device is not domain joined, a Microsoft account that has been granted administrative privileges on the device is required. When the administrator uses a Microsoft account to sign in, the clear key is removed, a recovery key is uploaded to the online Microsoft account, and a TPM protector is created. Should a device require the recovery key, the user will be guided to use an alternate device and navigate to a recovery key access URL to retrieve the recovery key by using his or her Microsoft account credentials.
- If the user uses a domain account to sign in, the clear key is not removed until the user joins the device to a domain and the recovery key is successfully backed up to Active Directory Domain Services (AD DS).
- Similar to signing in with a domain account, the clear key is removed when the user logs on to an Azure AD account on the device.

Used Disk Space Only Encryption

BitLocker in earlier Windows versions could take a long time to encrypt a drive, because it encrypted every byte on the volume (including parts that did not have data). That is still the most secure way to encrypt a drive, especially if a drive has previously contained confidential data that has since been moved or deleted, in which case traces of the confidential data could remain on portions of the drive marked as unused. To reduce encryption time, BitLocker in Windows 10 lets users choose to encrypt just their data. Depending on the amount of data on the drive, this option can reduce encryption time by more than 99%..

Exercise caution when encrypting only used space on an existing volume on which confidential data may have already been stored in an unencrypted state however, because those sectors can be recovered through disk-recovery tools until they are overwritten by new encrypted data. In contrast, encrypting only used space on a brand-new volume can significantly decrease deployment time without the security risk because all new data will be encrypted as it is written to the disk.

Encrypted hard drive support

SEDs have been available for years, but Microsoft couldn’t support their use with some earlier versions of Windows because the drives lacked important key management features. Microsoft worked with storage vendors to improve the hardware capabilities, and now BitLocker supports the next generation of SEDs, which are called encrypted hard drives. Encrypted hard drives provide onboard cryptographic capabilities to encrypt data on drives, which improves both drive and system performance by offloading cryptographic calculations from the PC’s processor to the drive itself and rapidly encrypting the drive by using dedicated, purpose-built hardware. If you plan to use whole-drive encryption with Windows 10, Microsoft recommends that you investigate hard drive manufacturers and models to determine whether any of their encrypted hard drives meet your security and budget requirements.

2.5.4 Malware resistance

Trusted Platform Module

A TPM is a tamper-resistant cryptographic module designed to enhance the security and privacy of computing platforms. The TPM is incorporated as a component in a trusted computing platform like a personal computer, tablet, or phone. The computing platform is specially designed to work with the TPM to support privacy and security scenarios that cannot be achieved through software alone. A proper implementation of a TPM as part of a trusted computing platform provides a hardware root of trust, meaning that the hardware behaves in a trusted way. For example, a key created in a TPM with the property that it can never be exported from the TPM really means the key cannot leave the TPM.

The close integration of a TPM with a platform increases the transparency of the boot process and supports device health scenarios by enabling reliable report of the software used to start a platform. The functionality a TPM provides includes:

- Cryptographic key management. Create, store, and permit the use of keys in defined ways.
- Safeguarding and reporting integrity measurements. Software used to boot the platform can be recorded in the TPM and used to establish trust in the software running on the platform.
- Prove a TPM is really a TPM. The TPM’s capabilities are so central to protecting privacy and security that a TPM needs to be able to differentiate itself from malware that masquerades as a TPM.

Among other functions, Windows 10 uses the TPM to protect the encryption keys for BitLocker volumes, virtual smart cards, certificates, and the many other keys that the TPM is used to generate. Windows 10 also uses the TPM to securely record and protect integrity-related measurements of select hardware and Windows boot components for the Measured Boot feature described later in this document. In this scenario, Measured Boot measures each component, from firmware up through the drivers, and then stores those measurements in the PC’s TPM.

Several improvements have been made in the TPM standard, the most notable of which is cryptographic agility. TPM 1.2 is restricted to a fixed set of encryption and hash algorithms. At the time the TPM 1.2 standard was created, in the early 2000s, these algorithms were considered cryptographically strong. Since then, advances in cryptographic algorithms and cryptanalysis attacks have increased expectations for stronger cryptography. TPM 2.0 supports additional algorithms that offer stronger cryptographic protection as well as the ability to plug in algorithms that may be preferred in certain geographies or industries.

Hardware security features and VBS

The core functionality and protection of Device Guard starts at the hardware level. Devices that have processors equipped with SLAT technologies and virtualization extensions, such as Intel VT x and AMD V, will be able to take advantage of a VBS environment that dramatically enhances Windows security by isolating critical Windows services from the operating system itself. This isolation is necessary, because you must assume that the operating system kernel will be compromised, and you need assurance that some processes will remain secure. Device Guard leverages VBS to isolate its Hypervisor Code Integrity (HVCI) service, which enables Device Guard to help protect kernel mode processes and drivers from vulnerability exploits and zero days. HVCI uses the processor’s IOMMU functionality to force all software running in kernel mode to safely allocate memory.

This means that after memory has been allocated, its state must be changed from writable to read only or execute only. By forcing memory into these states, it helps ensure that attacks are unable to inject malicious code into kernel mode processes and drivers through techniques such as buffer overruns or heap spraying. In the end, the VBS environment protects the Device Guard HVCI service from tampering even if the operating system’s kernel has been fully compromised, and HVCI protects kernel mode processes and drivers so that a compromise of this magnitude can't happen in the first place. Another Windows 10 feature that employs VBS is Credential Guard. Credential Guard protects credentials by running the Windows authentication service known as LSA, and then storing the user’s derived credentials (for example, NTLM hashes; Kerberos tickets) within the same VBS environment that Device Guard uses to protect its HVCI service.

By isolating the LSA service and the user’s derived credentials from both user mode and kernel mode, an attacker that has compromised the operating system core will still be unable to tamper with authentication or access derived credential data. Credential Guard prevents pass-the-hash and ticket types of attacks, which are central to the success of nearly every major network breach you’ve read about, which makes Credential Guard one of the most impactful and important features to deploy within your environments possible. In addition to these features, Microsoft recommends that you continue to maintain an enterprise antivirus solution for a well-rounded enterprise security portfolio.

Device Guard with Credential Guard

Although Credential Guard isn’t a feature within Device Guard, many organizations will likely deploy Credential Guard alongside Device Guard for additional protection against derived credential theft. Similar to virtualization-based protection of kernel mode through the Device Guard HVCI service, Credential Guard leverages hypervisor technology to protect the Windows authentication service (the LSA) and users’ derived credentials. This mitigation is targeted at preventing the use of pass-the-hash and pass-the-ticket techniques.

Because Credential Guard uses VBS, it is decisive in its ability to prevent pass-the-hash and pass-the-ticket attacks from occurring on Windows 10 devices. Microsoft recognizes, however, that most organizations will have a blend of Windows versions running in their environments. Mitigations for devices not capable of running Credential Guard on both the client side and the server side are available to help with this scenario. Microsoft will be releasing details to TechNet regarding these additional mitigations in the near future.

Data Execution Prevention

Malware depends on its ability to put a malicious payload into memory with the hope that it will be executed later, and ASLR will make that much more difficult. Data Execution Prevention (DEP) does exactly that, by substantially reducing the range of memory that malicious code can use for its benefit. DEP uses the No execute bit on modern CPUs to mark blocks of memory as read-only so that those blocks can't be used to execute malicious code that may be inserted within through a vulnerability exploit. Because of the importance of DEP, users cannot install Windows 10 on a computer that does not have DEP capability. Fortunately, most processors released since the mid-2000s support DEP.

Control Flow Guard

When applications are loaded into memory, they are allocated space based on the size of the code, requested memory, and other factors. When an application begins to execute code, it calls additional code located in other memory addresses. The relationships between the code locations are well known—they are written in the code itself but previous to Windows 10, the flow between these locations was not enforced, which gives attackers the opportunity to change the flow to meet their needs. In other words, an application exploit takes advantage of this behavior by running code that the application may not typically run. This kind of threat is mitigated in Windows 10 through the Control Flow Guard (CFG) feature.

When a trusted application that was compiled to use CFG calls code, CFG verifies that the code location called is trusted for execution. If the location is not trusted, the application is immediately terminated as a potential security risk. An administrator cannot configure CFG; rather, an application developer can take advantage of CFG by configuring it when the application is compiled. Administrators should consider asking application developers and software vendors to deliver trustworthy Windows applications compiled with CFG enabled. Of course, browsers are a key entry point for attacks; thus Microsoft Edge, IE, and other Windows features take full advantage of CFG.

Protected Processes

No computer is immune to malware, however. Despite all the best preventative controls, malware can eventually find a way to infect any operating system or hardware platform. So, although prevention with a defense-in-depth strategy is important, it cannot be the only type of malware control. The key security scenario is to assume that malware is running on a system but to limit what it can do. Windows 10 has security controls and design features in place to reduce compromise from existing malware infections. Protected Processes is one such feature. With Protected Processes, Windows 10 prevents untrusted processes from interacting or tampering with those that have been specially signed. Protected Processes defines levels of trust for processes. Less trusted processes are prevented from interacting with, and therefore attacking, more trusted processes.

Windows 10 uses Protected Processes more broadly across the operating system and, for the first time, you can put antimalware solutions into the protected process space, which helps make the system and antimalware solutions less susceptible to tampering by malware that does manage to get on the system.

Chapter 3- Comparison of Windows

3.1 Differences of Windows

EMET proved useful for a couple of reasons. First, it allowed us to interrupt and disrupt many of the common exploit kits employed by attackers at the time without waiting for the next Windows release.Second, we were able to use EMET as a place to assess new features, which directly led to many security innovations in Windows 7, 8, 8.1, and 10. But EMET has serious limits as well – precisely because it is not an integrated part of the operating system. First, many of EMET’s features were not developed as robust security solutions. As such, while they blocked techniques that exploits used in the past, they were not designed to offer real durable protection against exploits over time. Not surprisingly, one can find well-publicized, often trivial bypasses, readily available online to circumvent EMET.

Second, to accomplish its tasks, EMET hooks into low-level areas of the operating system in ways they weren’t originally designed. This has caused serious side-effects in both performance and reliability of the system and the applications running on it. This presents an ongoing problem for customers since every OS or application update can trigger performance and reliability issues due to incompatibility with EMET. By detecting and preventing the buffer overflows and memory corruption vulnerabilities often exploited in zero-day attacks, the free EMET tool has often been recommended by Microsoft in security bulletins as a way of mitigating against exploits while they work on a proper patch. Through technologies such as Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR) and pinning rules to validate digitally signed certificates while browsing, EMET can act as part of your layered defence. In short, you don’t get rid of other technologies such as anti-virus software, but EMET helps sandbox Windows and Windows apps to make them harder to exploit, and prevent unpatched vulnerabilities from being successfully weaponised.

Although primarily targeted at system administrators responsible for securing Windows PCs on larger networks because of the ability to manage and apply group policies, there’s actually nothing to prevent you from also running EMET on your home computer if you wish to block common memory exploitation techniques and lock down applications.

Windows 10 Cannot Protect Insecure Applications Like EMET Can

Recently, Microsoft published a blog post with two main points: Microsoft EMET will no longer support EMET after July 31, 2018, and Windows 10 provides protections that make EMET unnecessary.

System-Wide Protection

- Data Execution Prevention (DEP)
- Structured Exception Handler Overwrite Protection (SEHOP)
- Address Space Layout Randomization (ASLR)
- Certificate Trust (Pinning)
- Block Untrusted Fonts (Fonts)

The system-wide DEP, SEHOP, and ASLR settings in EMET are provided by the Windows operating system itself. That is, the benefit of EMET for these settings is simply that it acts as a unified GUI application to make these changes in your system.

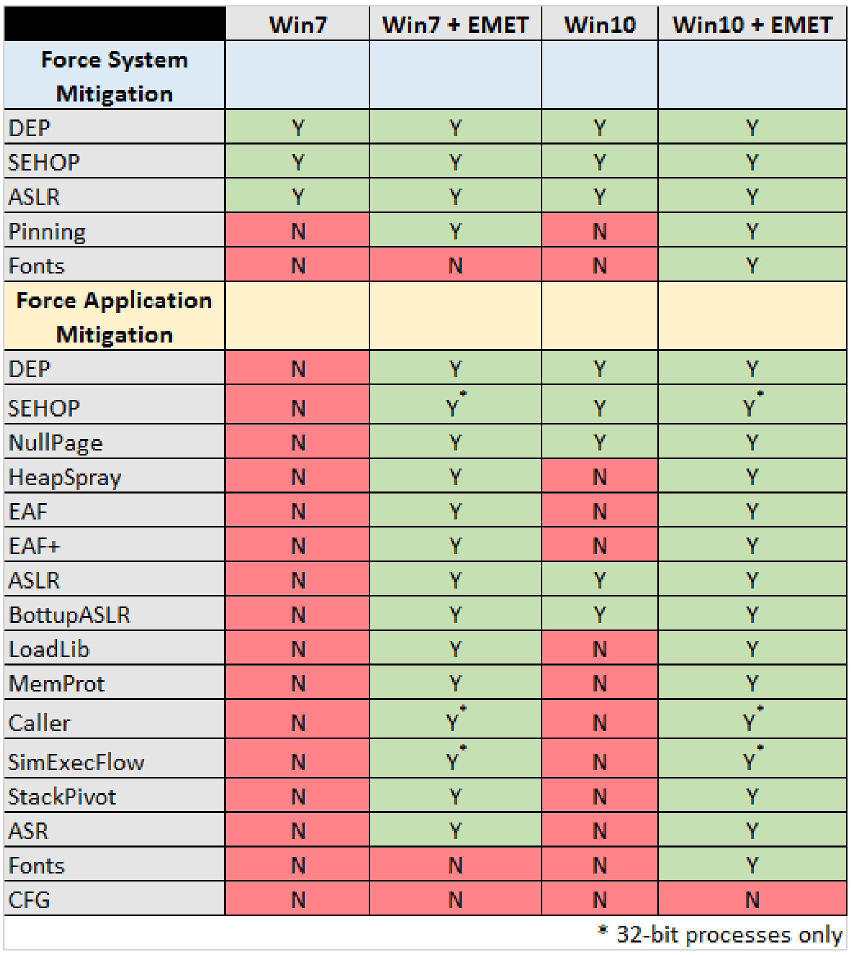
Application-Specific Protection

- Data Execution Prevention (DEP)

- Structured Exception Handler Overwrite Protection (SEHOP)
- Null Page Allocation (NullPage)
- Heapspray Allocations (HeapSpray)
- Export Address Table Access Filtering (EAF)
- Export Address Table Access Filtering Plus (EAF+)
- Mandatory Address Space Layout Randomization (ASLR)
- Bottom-Up Randomization (BottomUpASLR)
- ROP MitigationsMemPrSimExec
- FlowAttack Surface Reduction (ASR)

The purpose of the table below is to draw attention to the application specific mitigations that EMET provides, but Windows 10 does not provide. Windows 10 includes a number of extra system-level mitigations that Windows 7 with EMET cannot provide. This is where EMET can help and why so many of the security professionals. EMET can be used to provide protection for individual applications in your environment. For example, EMET on a system it helped to immediately identify the running processes that were not using DEP. One of these applications allows the user to receive files from a scanner connected to the network.

EMET allowed to enable DEP on this application without requiring recompiling it or getting a new version of the application from the vendor that developed it. This is especially handy for deploying mitigations on older software that was written before the mitigations were available and where source code is not currently available. If you don't use both of those, then there are many powerful techniques for exploiting a buffer overrun. For instance, DEP alone can be defeated using return-oriented computing; and ASLR alone can be defeated using heap spraying and repeated attempts. However, if the target uses both ASLR + DEP, exploitation becomes significantly harder.



	Win7	Win7 + EMET	Win10	Win10 + EMET
Force System Mitigation				
DEP	Y	Y	Y	Y
SEHOP	Y	Y	Y	Y
ASLR	Y	Y	Y	Y
Pinning	N	Y	N	Y
Fonts	N	N	N	Y
Force Application Mitigation				
DEP	N	Y	Y	Y
SEHOP	N	Y*	Y	Y*
NullPage	N	Y	Y	Y
HeapSpray	N	Y	N	Y
EAF	N	Y	N	Y
EAF+	N	Y	N	Y
ASLR	N	Y	Y	Y
BottupASLR	N	Y	Y	Y
LoadLib	N	Y	N	Y
MemProt	N	Y	N	Y
Caller	N	Y*	N	Y*
SimExecFlow	N	Y*	N	Y*
StackPivot	N	Y	N	Y
ASR	N	Y	N	Y
Fonts	N	N	N	Y
CFG	N	N	N	N

* 32-bit processes only

Figure 4

The techniques mentioned above are not sufficient to defeat ASLR + DEP. ASLR + DEP are like a one-two punch that make the attacker's life much harder. Defeating the combination of ASLR + DEP is not impossible, but it takes much more cleverness.

Canary

Stack canaries work by modifying every function's prologue and epilogue regions to place and check a value on the stack respectively. As such, if a stack buffer is overwritten during a memory copy operation, the error is noticed *before* execution returns from the copy function. When this happens, an exception is raised, which is passed back up the exception handler hierarchy until it finally hits the OS's default exception handler. If you can overwrite an existing exception handler structure in the stack, you can make it point to your own code. This is a Structured Exception Handling (SEH) exploit, and it allows you to completely skip the canary check.

DEP / NX

DEP and NX essentially mark important structures in memory as non-executable, and force hardware-level exceptions if you try to execute those memory regions. This makes normal stack buffer overflows where you set **eip** to **esp+offset** and immediately run your shellcode impossible, because the stack is non-executable. Bypassing DEP and NX requires a cool trick called Return-Oriented Programming.

ROP essentially involves finding existing snippets of code from the program (called gadgets) and jumping to them, such that you produce a desired outcome. Since the code is part of legitimate executable memory, DEP and NX don't matter. These gadgets are chained together via the stack, which contains your exploit payload. Each entry in the stack corresponds to the address of the next ROP gadget. So that the ret will jump to the next address on the stack after executing the instructions, thus chaining the gadgets together. Often additional values have to be placed on the stack in order to successfully complete a chain, due to instructions that would otherwise get in the way.

The trick is to chain these ROPs together in order to call a memory protection function such as VirtualProtect, which is then used to make the stack executable, so your shellcode can run, via an **jmp esp** or equivalent gadget.

ASLR

There are a few ways to bypass ASLR:

- Direct RET overwrite - Often processes with ASLR will still load non-ASLR modules, allowing you to just run your shellcode via a **jmp esp**.
- Partial EIP overwrite - Only overwrite part of EIP, or use a reliable information disclosure in the stack to find what the real EIP should be, then use it to calculate your target. We still need a non-ASLR module for this though.
- NOP spray - Create a big block of NOPs to increase chance of jump landing on legit memory. Difficult, but possible even when all modules are ASLR-enabled. Won't work if DEP is switched on though.

3.1.2 Windows 8 is more secure than Windows 7

Early Launch Anti-Malware

In Windows 8, antivirus products can start earlier in the boot-up process to scan the system’s drivers for malware. This helps protect against rootkits that start before the antivirus program and hide from it. Windows Defender starts earlier in the boot process out-of-the-box, and third-party antivirus vendors can also add the Early-Launch Anti-Malware (ELAM) feature to their products.

Table 3: Compare Windows 7-8

Windows 7	Windows 8
When BitLocker is used with a PIN to protect startup, PCs such as servers cannot be restarted remotely.	**Network Unlock allows PCs to start automatically when connected to the internal network.**

Enabling BitLocker can make the provisioning process take several hours.	**Windows 8 allows users with standard privileges to change their BitLocker PIN or passwo**
No support for using BitLocker with Self-Encrypting Drives (SEDs).	**BitLocker pre-provisioning and Used Space Only encryption allow BitLocker to be quickly enabled on new computers.**
Administrators have to use separate tools to manage encrypted hard drives.	**BitLocker supports offloading encryption to encrypted hard drives.**
Encrypting a new flash drive can take more than 20 minutes.	**BitLocker To Go's Used Space Only encryption allows users to encrypt drives in seconds.**

Table 4: Compare Windows 7-8

Multi-factor solutions are often cumbersome and costly to deploy.	Microsoft Passport is an easy to use and easy to deploy, multi-factor, password alternative.
Phishing attacks on your users' passwords are increasingly successful.	Windows Hello uses biometrics to provide a more secure way of accessing your device, Microsoft Passport, apps, data, and online resources.
Pass the Hash attacks enable attackers to steal identities, traverse across networks, and evade detection.	Microsoft Azure Active Directory provides a comprehensive identity and access management solution for the cloud.
BitLocker offers optionally configurable disk encryption.	BitLocker is much improved, is highly manageable, and can be automatically provisioned on most new devices.
Data loss prevention (DLP) requires the use of additional software and frequently third-party capability.	Enterprise Data Protection addresses the needs for DLP, includes a deeply integrated data separation and containerization solution, and provides encryption at the file level.
DLP solutions often compromise the user experience in the interest of security, resulting in low adoption and varying experience between the desktop and	Enterprise Data Protection provides a seamless user experience across mobile devices and the desktop, and is integrated with Azure Active Directory and Rights

mobile devices.	Management Services.
All apps are trusted until they’re determined to be a threat or are explicitly blocked.	Device Guard offers protection on the desktop that is similar to lockdown on a mobile platform (full app lockdown).
With more than 300,000 new threats per day, blocking them through detection (block on known bad) is a losing battle.	With Device Guard, an application must prove itself to be trustworthy before it can be run.
Windows provides a series of defense solutions, but too many malware threats impact users before detection-based antivirus solutions can catch up.	Device Guard will be the most disruptive malware-resistance capability Microsoft has ever shipped in the desktop.
Platform security is based entirely on what software can do on its own, and once infected there is no assurance that system defenses can perform their function and remain tamper free.	Hardware-based security and the level of trust it offers helps to maintain and validate hardware and system integrity.

Table 5: Windows 10 Protections

Threat	**Windows 10 mitigation**
"Man in the middle" attacks, when an attacker reroutes communications between two users through the attacker's computer without the knowledge of the two communicating users	Client connections to the Active Directory Domain Services default SYSVOL and NETLOGON shares on domain controllers now require SMB signing and mutual authentication (such as Kerberos).
Firmware bootkits replace the firmware with malware.	All certified PCs include a UEFI with Secure Boot, which requires signed firmware for updates to UEFI and Option ROMs.
Bootkits start malware before Windows starts.	UEFI Secure Boot verifies Windows bootloader integrity to ensure that no malicious operating system can start before Windows.
System or driver rootkits start kernel-level malware while Windows is starting, before Windows Defender and antimalware solutions can start.	Windows Trusted Boot verifies Windows boot components; Microsoft drivers; and the Early Launch Antimalware (ELAM) antimalware driver, which verifies non-Microsoft drivers. Measured Boot runs in parallel with Trusted Boot and can provide information to a remote server that verifies the boot state of the device to help ensure Trusted Boot and other boot components successfully checked the system.
User-level malware exploits a vulnerability in the system or an application and owns the device.	Improvements to address space layout randomization (ASLR), Data Execution Prevention (DEP), the heap architecture, and memory-management algorithms reduce the likelihood that vulnerabilities can enable Protected Processes isolates non-trusted processes from each other and from sensitive operating system components.VBS, built on top of Microsoft Hyper-V, protects sensitive Windows processes from the Windows

Threat	Windows 10 mitigation
	operating system by isolating them from user mode processes and the Windows kernel. Configurable code integrity enforces administrative policies to select exactly which applications are allowed to run in user mode. No other applications are permitted to run.
Users download dangerous software (for example, a seemingly legitimate application with an embedded Trojan horse) and run it without knowledge of the risk.	The SmartScreen Application Reputation feature is part of the core operating system; Microsoft Edge and Internet Explorer can use this feature either to warn users or to block users from downloading or running potentially malicious software.

3.1.3 Heap mitigation techniques

The hardening changes that have been made to the Windows heap manager generally fall into two categories: metadata protection and non-determinism. Metadata protection changes focus on protecting the integrity of various data structures that are used internally by the heap manager. These changes are useful because the majority of public exploitation techniques have traditionally relied on the corruption of one or more heap data structure. On the other hand, non-determinism changes focus on making the state of the heap unpredictable, which has a direct impact on the probability that an exploit will succeed.

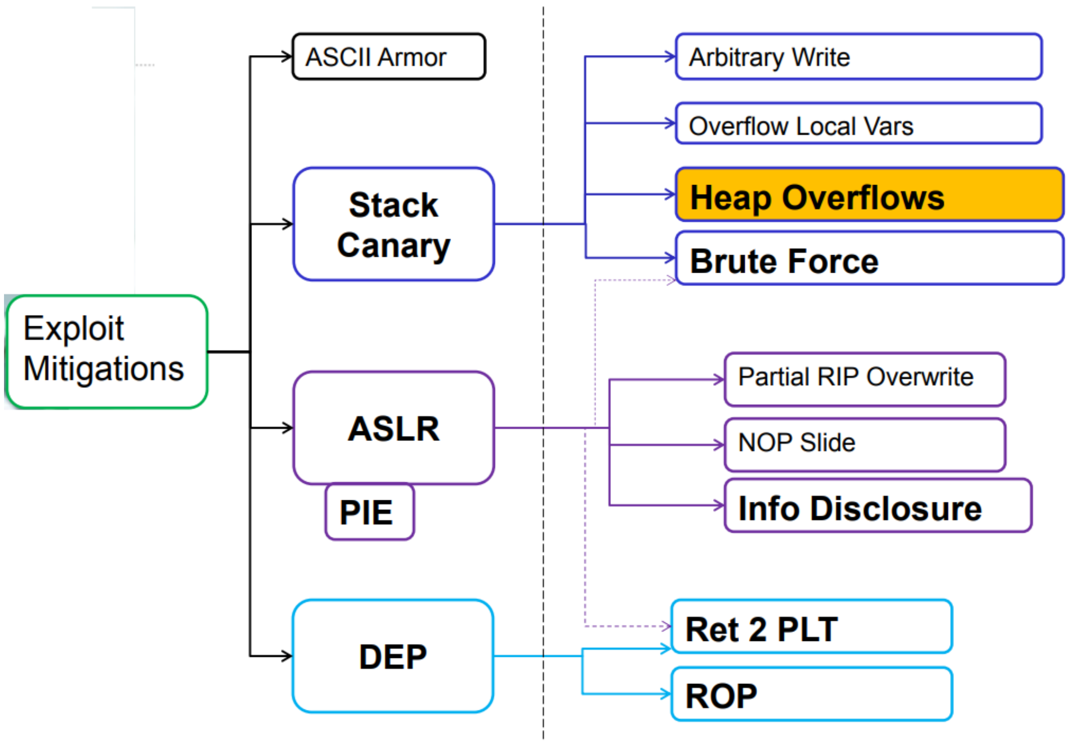


Figure 5

Security was a top priority during Windows Vista development. In Windows XP, every user is set up as an administrator by default. As a result, most home users ran all their software with Administrator access. However, this left most users unwittingly open to potential security threats, such as hacking and malware downloads. A large amount of existing software doesn’t run well as a standard user, due to developers not implementing the principle of least privilege in their design and testing. For example, many poorly written applications often assume incorrectly that they will have read and write access to the entire file system and system registry. Denying such an application access to any of these assumed rights can cause the application to fail.

Table 6: Compare Windows XP-Vista-7

Windows Vista and Windows 7	**Windows XP**

The heap manager in Windows Vista, Windows Server 2008, and Windows 7 expanded on the hardening work that went into Windows XP SP2 and Windows Server 2003 SP1 by incorporating a number of additional security improvements. These improvements are enabled by default (with the exception of termination on heap corruption) and include: **Removal of commonly targeted data structures:** Heap data structures such as look aside lists and array lists, which have been targeted by multiple exploitation techniques, have been removed. Lookaside lists have been replaced by the Low Fragmentation Heap (LFH).	The first set of heap hardening changes were released with Windows XP SP2 and Windows Server 2003 SP1.These changes included: **Safe unlinking** A verification check that occurs during free chunk unlinking which makes sure that the list entry stored in a free chunk is a valid doubly linked list entry (by checking E->F->B == E->B->F == E where E is the free chunk list entry). This prevents exploitation techniques that rely on using the unlink operation performed during the coalescing of free chunks to write an arbitrary value to an arbitrary address in memory.
Heap entry metadata randomization The header associated with each heap entry is XORd with a random value in order to protect the integrity of the metadata. The heap manager then unpacks and verifies the integrity of each heap entry prior to operating on it.	**Heap entry header cookie** An 8-bit random value was added to the header of each heap entry which is validated when a heap entry is freed. This makes it possible to detect corruption when a chunk is deallocated.
Randomized heap base address The base address of a heap region is randomized as part of the overall Address Space Layout Randomization (ASLR) implementation and has 5 bits of entropy. This is designed to make the address of heap data structures	

and heap allocations unpredictable to an attacker.	
Expanded role of heap header cookie The 8-bit random value that is associated with the header of each heap entry has had its scope extended to enable integrity checking of more fields. The cookie's value is also verified in many more places (rather than only checking at the time that a heap entry is freed).	
Function pointer encoding Function pointers in heap data structures are encoded with a random value to prevent them from being replaced with an untrusted value.	
Termination on heap corruption If enabled, any detected corruption of a heap data structure will lead to immediate process termination. This is the default for most built-in Windows applications, and can be enabled dynamically by third parties. If disabled, corruption errors are ignored and the application is allowed to continue executing.	

Sometimes, a person logged on as an standard user under Windows XP can't perform user-specific tasks such as changing the system clock and calendar, changing the computer's time zone, or changing the computer's power management settings due to so-called "LUA bugs". Windows Vista also includes Windows Defender, a spyware scanning and removal tool that is also available for Windows XP for free. Enterprise and Ultimate editions of Windows Vista include BitLocker Drive Encryption, which aims to help protect data in the case of stolen devices. Vista implements address space layout randomization (ASLR) that makes it considerably harder for malicious code to exploit return to attacks than on previous versions of Windows, particularly on 64-bit systems. Furthermore, Vista implements heap management enhancements that make it much more difficult to carry out buffer-overflow attacks.

Table 7: Differences in System and App on Windows

	Win7	Win7+EMET	Win10 1607 Anniversary Update	Win10 1703 Update
Default Enabled System Mitigation				
Kernel pool hardening [	Y	No Change	Y	Y
Kernel ASLR (images) [	Y	No Change	Y	Y
Fonts (usermode appcontainer)	N	N	Y	Y
NTVDM disabled	N	N	Y	Y
Kernel ASLR (full)	N	N	Y	Y
Kernel DEP	N	N	Y	Y
Kernel pool hardening (extended)	N	N	Y	Y
SMEP	N	N	Y	Y
Global safe unlinking	N	N	Y	Y
Improved ASLR entropy	N	N	Y	Y
Opt-In System Mitigation				
DEP	Y	Y	Y	Y
SEHOP	Y	Y	Y	Y
ASLR	Y	Y	Y	Y
Pinning	N	Y	Y	Y
Fonts (block untrusted)	N	N	Y	Y
VBS – HyperGuard (protect MSR/SMEP)	N	N	Y	Y
VBS – HVCI (kernel CI)	N	N	Y	Y
VBS – Credential Guard	N	N	Y	Y
VBS – Device Guard	N	N	Y	Y
SecureBoot	N	N	Y	Y
Default Enabled Application Mitigation				
Heap metadata hardening	Y	Y	Y	Y
Heap metadata hardening	N	N	Y	Y
Heap allocation randomization	N	N	Y	Y
Heap guard pages	N	N	Y	Y
AppContainer symbolic link hardening	N	N	Y	Y
Opt-In Application Mitigation				
SEHOP	Y	Y	Y	Y
DEP	N	Y	Y	Y
NullPage	N	Y	Y	Y

Force ASLR	N	Y	Y	Y
BottomupASLR	N	Y	Y	Y
LoadLib (Image Load Restriction)	N	Y	Y	Y
MemProt (Dynamic Code Restriction)	N	Y	Y	Y
Fonts (block untrusted)	N	N	Y	Y
Child Process Restriction	N	N	Y	Y
Code Integrity Restriction	N	N	Y	Y
Win32k System Call Disable Restriction	N	N	Y	Y
High Entropy ASLR	N	N	Y	Y
Strict handle checks	N	N	Y	Y
Extension point disable	N	N	Y	Y
Heap terminate on corruption	N	N	Y	Y
ASR	N	Y	N	Y
HeapSpray	N	Y	N	N

3.2 Forensics for Windows

Digital forensics, also known as computer and network forensics, is the application of science to the identification, collection, examination, and analysis of data while preserving the integrity of the information and maintaining a strict chain of custody for the data. The forensic analysis goal is to gain a better understanding of an event of interest by finding and analyzing the facts related to that event. Forensics may be needed in many different situations, such as evidence collection for legal proceedings and internal disciplinary actions, and handling of malware incidents and unusual operational problems.

The world we live in today is a technologically advanced world. While on one hand, commercialization of IT (Information technology) revolutionized our modern-day lifestyle, it has raised a big question mark about the confidentiality and privacy of the information shared and managed using advanced means of communication. As computer technology continues to evolve, the task of managing and handling private and sensitive information is becoming more and more challenging with each passing day. Increased rates of cyber-crime leading to unsolicited invasions of privacy have resulted in the emergence of a new field of computer science known as cyber forensics. With the increasing demand of computer security in recent times, it has become more important than ever to understand the digital forensic technology.

What is Digital Forensics/Cyber Forensics?

Also known as cyber forensics, computer forensics involve the application of acquiring and analyzing digital information (as a part of a structured investigation) to be used as evidence in the court of law.

Digital Forensics-Primary Goals

The primary goal of Digital Forensics is to carry out an organized and structured investigation in order to preserve, identify, extract, document and interpret digital information that is then utilized to prevent, detect and solve cyber incidents.

A typical forensic investigation consists of the following main steps:

1. Preserving the data.
2. Acquiring the data.
3. Authenticating the data.
4. Analyzing the data.

1. Preserving and acquiring the data-The first and foremost step of a digital forensic investigation is to preserve and acquire the data from a computer. This step involves creating a bit by bit copy of the hard drive data.
2. Authenticating the data- The next process involves verifying the data seized. To ensure that the acquired data is an exact copy of the contents of the hard drive, the md5/sha1 of the original and copied data are checked and matched.
3. Analyzing the data-This is perhaps the most important part of the investigation process which involves careful examination and analysis of the data using forensic tools.

The process mainly involves:

- Recovering deleted files /Data Recovery
- Tracking or identifying hacking activities

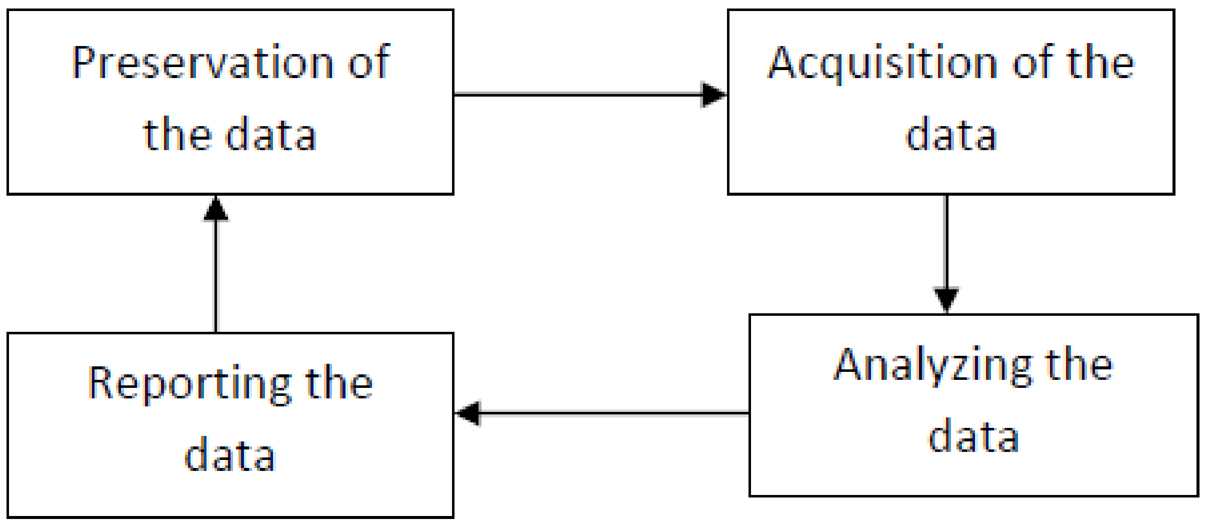


Figure 6

The transformation of the analog world into a digital world has raised new challenges and opportunities for technology lovers. New forensic challenges arise with the introduction of newly released and latest operating systems. While on one hand, these newly released versions of Windows are aimed at making things easier for users, many of the functions performed by your operating system for your convenience can actually be used against you.

3.2.1 Computer Forensics

The impact forensic science has had on countless criminal investigations and trials make it a crucial part of law enforcement. Therefore, it is necessary to continue advancing the forensic science to meet the increasing demand of law enforcement against cyber-crime. In cybercrime investigations, the crime scene can consist of one or more computers perhaps spanning one or more computer networks. A cyber-criminal may affect a system locally or remotely. A local attacker may leave physical evidence at the scene such as witnesses or fingerprints in addition to electronic evidence. Via the Internet, a remote attacker can penetrate other systems connected to the Internet from anywhere in the world, leaving only electronic evidence. In either case, digital forensic evidence can be gathered from the criminal's computer, the victim's computer, or both.

This digital evidence can be broadly categorized in two ways, non-volatile and volatile. Non-volatile electronic evidence can be recovered after a system is powered down and is found on hard drives, USB flash drives, and floppy disks. It is in non-volatile memory where most of the electronic evidence originates. System logs, network logs, malicious code, corrupted files, emails, internet browser cached files and history, and deleted files are all forensic evidence stored in non-volatile memory. Network logs may contain TCP session logs indicating the source IP address from where the attack originated. The malicious code may be analyzed to determine exactly what the attacker did to the system. Analysis of the disk drive’s file system can lead to the recovery of deleted files, which may contain further evidence. Electronic evidence gathered from non-volatile memory can be used to determine how and when a system was infiltrated, what files were corrupted and how, and how much damage, if any, was done to the system.

As for the criminal’s computer, email and browser history and cache can prove the criminal’s intent, expose any accomplices, and even give further evidence of how an attack, if any, occurred. However, a careful cyber-criminal may have permanently erased any incriminating evidence from non-volatile memory, thereby making its recovery impossible. In the case of an infiltrated computer running malicious code, there is other evidence that can be useful. The other type of electronic evidence is in volatile memory.

Unlike data stored on hard drives, electronic evidence found in main memory disappears once power is removed from the system. Information about each running process, such as create times, exit times, open files, executing code, and child process are stored in main memory. This type of evidence is useful if a malicious program is running or another program has been corrupted on a live system. Unlike the non-volatile memory, this evidence cannot be erased from memory as long as malicious code is running. Additionally, trusted programs may be used to gather data from a live system such as open network ports, established network connections, logged on users, and list of running processes.

As with other forms of forensic evidence, special tools are necessary to analyze computer crime scenes. At present, few computer forensic tools exist that are capable of performing anything other than a rudimentary analysis of non-volatile evidence gathered from running computer systems.

3.2.2 Digital Forensics and Windows-The Windows Artifacts.

The average user is mostly unaware of the fact that their newly upgraded operating system is leaving tracks of their activity. It is essential for users to know that valuable pieces of sensitive and confidential information is stored in Windows Artifacts. These artifacts can be used to recreate and restore the account history of a particular user.

Some of the artifacts of Windows operating system include:

1) Root user Folder
2) Desktop
3) Pinned files
4) Recycle Bin Artifacts
5) Registry Artifacts App Data Artifacts
6) Favorites Artifacts
7) Send to Artifacts
8) Swap Files Artifacts
9) Thumb Cache artifacts
10) HKey Class Root Artifacts
11) Cookies Artifacts
12) Program files Artifacts
13) Meta Data Artifacts
14) My Documents Artifacts
15) Recent Folder Artifacts
16) Restore Points Artifacts
17) Print Spooler Artifacts
18) Logo Artifacts
19) Start menu Artifacts,
20) Jump lists

Information collected from any of these artifacts can be used to recreate the account history of a user. To gain a better understanding of how these artifacts can be used to access or retrieve valuable information, it is essential to briefly discuss some of the most important Artifacts of Windows 7.

1. <u>Root User Folder artifacts</u>

The Root User Folder gives access to the complete operating system. The Root User reserves the right to delete and modify files on the operating system besides having the rights to generate new users and award them some rights. Nonetheless, these rights cannot exceed the rights of a root user.

The Windows Folder is specified by %SYSTEMROOT%. The Folder can be accessed through Start\Run\%SYSTEMROOT%\System32.

2. Desktop Artifacts

All the files present on the desktop of a user are stored in the desktop folder of the operating system. Typically, the desktop is populated either,

– By the user, or

– By programs that automatically create files and place them on the desktop.

The Desktop can be accessed using the following link;

C:\USERS\username\desktop

3. Pinned Files/Jump Lists Artifacts

Pinned Files or Jump lists are a relatively new feature introduced in Windows 7 released by Microsoft. Using the Jump lists all the pinned files can be accessed. Additionally, these lists also maintain a record of recently or last visited files relative to a particular software. Pinned files can be accessed from the jump list using the following link,

C:\Users\username\AppData\Roaming\Microsoft\InternetExplorer\QuickLaunch\UserPinned\TaskBar.

4. Recycle Bin Artifacts

The Recycle Bin stores the recently deleted files temporarily. These files can be restored easily. You can only view the Recycle Folder after un-checking the hide\protect system files option using the following link; **C:\$recycle.bin**

5. Registry Artifacts

Registry is the location where the configuration information of Windows is kept and stored. It can be used to obtain information related to historical and current use of applications in addition to obtaining valuable pieces of information about option preferences and system settings. It can be accessed using the following link;

NTUSER.DAT\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\WordWheelQuerry

6. App Data Artifacts

Application data or App data is a junction designed to provide backward compatibility. A junction can roughly be defined as a shortcut that serves to redirect programs and files to different locations. All the information related to settings configuration (of various apps) is stored in this folder. Furthermore, information related to the Windows address book and recently accessed files are also stored in this folder. The junction can be accessed through:

C: User\ (username)\AppData\Roamingfolder

7. Favorite Artifacts

This folder contains valuable bits of information related to Windows Explorer and Internet Explorer favorites. The folder can be accessed using the following link;

C:\USERS\username\favorites.

8. Send to Artifacts

The Send to folder stores information pertaining to shortcuts to different locations, and other software apps on the operating system of your computer. These shortcuts serve as destination points. Using these destination points a file can be sent or activated. Furthermore, these points can also be modified as per your convenience. The Send to folder can be accessed using the following link;

C:\Users\username \AppData\Roaming\Microsoft\Windows\SendTo

9. Swap Files Artifacts

Page Files or Swap files are the memory files of your computer that aid in expanding the memory of your computer. These files are not visible and are hidden by default settings. To view these files, following link can be used;
MyComputer>Properties>Taskmenu>AdvancedSystemSettings>Advancedtab>Performance>Settings>

Performance options dialogue box>Advanced tab>Change.

10. Thumbs Cache Artifacts

Thumbs.db files are files that are stored in every directory on the Windows systems that includes thumbnails. These are default files (created by default) and store valuable information that is not available elsewhere. The file is created locally amongst the images. The location where cache is stored is as follows;
C:\Users\Username\AppData\Local\Microsoft\Windows\Explorer

The display can be stopped by a user by checking on the ‘Always show icon, not thumbnails’ from the list of Folder options.

11. HKey Class Root Artifacts

The HKey Class Root or simply HKCR key contains sensitive information about different file name extensions in addition to containing information related to COM class registration. Furthermore, it is designed to be compatible with the 16-bit Window registry.

HKEY _LOCAL_MACHINE and HKEY_CURRENT_USER key both store valuables information related to file name extensions and class registration.

HKEY_LOCAL_MACHINE\Software\Classes: These key stores all the information pertaining to different users using the system.

The HKEY_CURRENT_USER\Software\Classes: On the other hand, these key stores information pertaining to the interactive user.

12. Cookies Artifacts

A number of websites store information on your computer in the form of cookies. Cookies can roughly be defined as small text files containing information related to preferences and configuration of a particular user.

These files can be accessed using the following link;

C: User\(username)\AppData\Roaming folder\ Microsoft\Windows\Cookies.

13. Program Files Artifacts

Windows 7 consists of two Program files folders including;
1. C:\program files
2. C:\Program files (x86)

These folders are designed to be compatible for 32 bits and 64 bits version of Windows 7. The first one is compatible with the 64 bit version of Windows 7, whereas, the second one is compatible with 32 bit version of Windows 7.

14. Meta Data Artifacts

Meta Data simply refers to information related to data itself. Using the metadata artifacts, valuable strings of file information can be obtained and can be used as evidence in digital forensic investigation.

15. Restore Points Artifacts

Windows & gives its users the option of restoring points thereby creating the image of your system. This essentially helps in providing users with an option to revert back to the point when the system was working perfectly in case of fatal system errors. This system image also contains the drives that are required by your operating system to run in addition to including program settings, system settings and file settings.

16. <u>My Documents Artifacts</u>

My Documents contains all the information related to files that have been created by users themselves. Usually when a program is installed on a system, the information is stored in this folder. It is also known as the primary storage space meant for storing all the key information. The folder can be accessed through;
C:\\Users\username\MyDocuments.

17. <u>Start Menu Artifacts</u>

The traditional Start menu has been replaced by Start in Windows 7. Using software like classic shell, it is absolutely possible to get the menu back. In Windows 7, the right column of the start (new version of start menu), links to respective libraries are shown instead of folders.

18. <u>Logo Artifacts</u>

The Logos included in the Windows 7 Operating System include valuable information pertaining to application events information, security related events information, setup event information, forwarded event information, and application events information.

19. <u>Print Spooler Artifacts</u>

Print Spooler is a software program responsible for organizing all the print jobs that have been sent to the print server or the computer printer. In essence all the print related information is stored in this folder.
C:\\Window\System32\Spool\Printers.

20. <u>Recent Folder Artifacts</u>

The Recent Folder stores links of the recently accessed or opened files by a specific user. The folder can be accessed by using the following link;

C:\Users\username\AppData\Roaming\Microsoft\Windows\Recent.

Windows Forensics- Analysis of Windows Artifacts

Analysis of Windows artifacts is perhaps the most crucial and important step of the investigation process that requires attention to detail. The following flowchart depicts a typical windows artifact analysis for the collection of evidence.

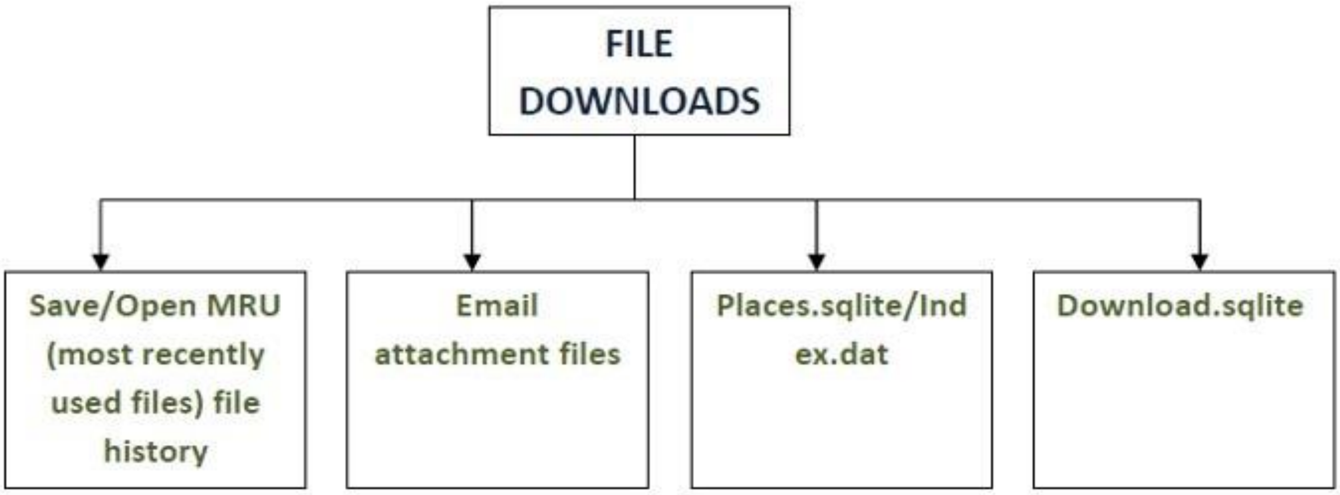


Figure 7

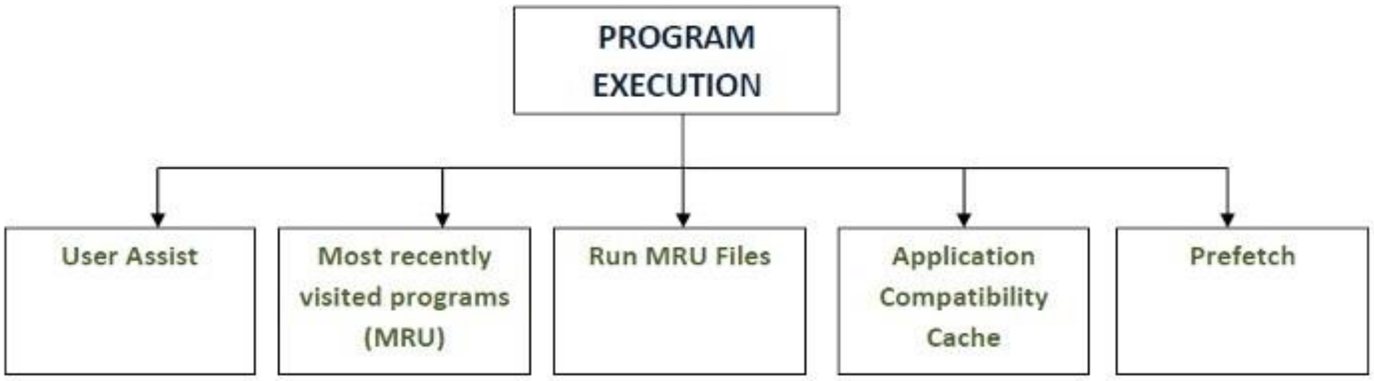

PROGRAM EXECUTION
User Assist
Most recently visited programs (MRU)
Run MRU Files
Application Compatibility Cache
Prefetch

Figure 8

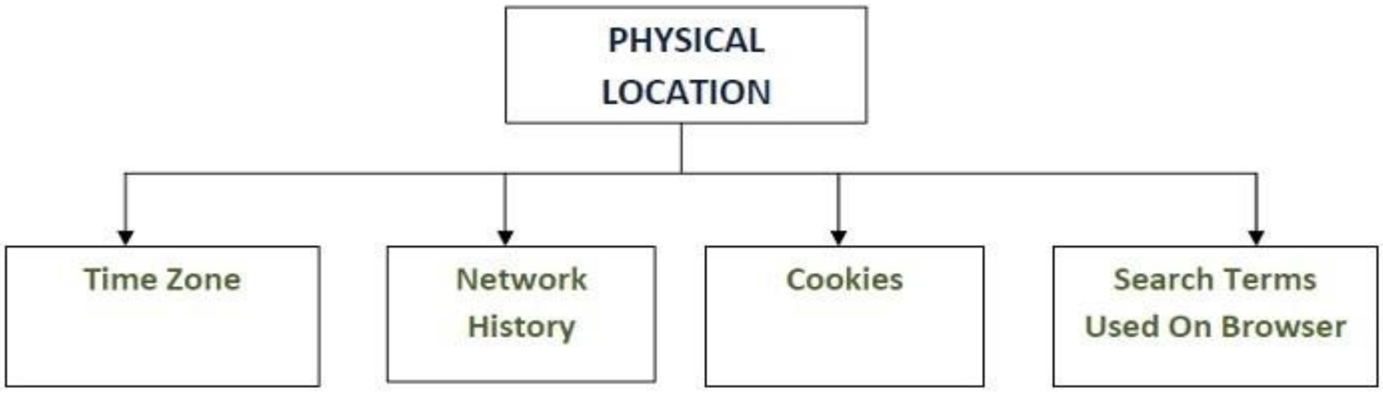


Figure 9

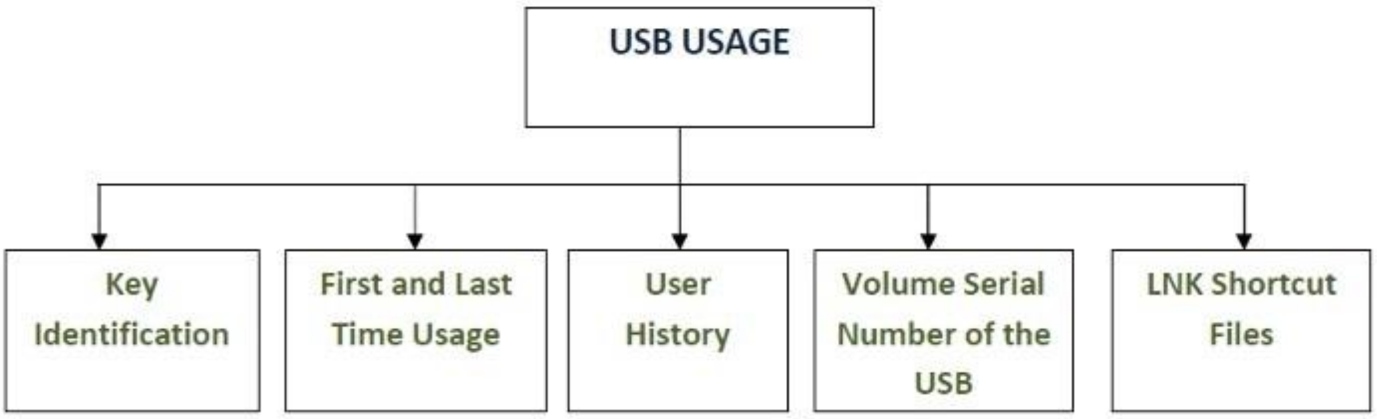


Figure 10

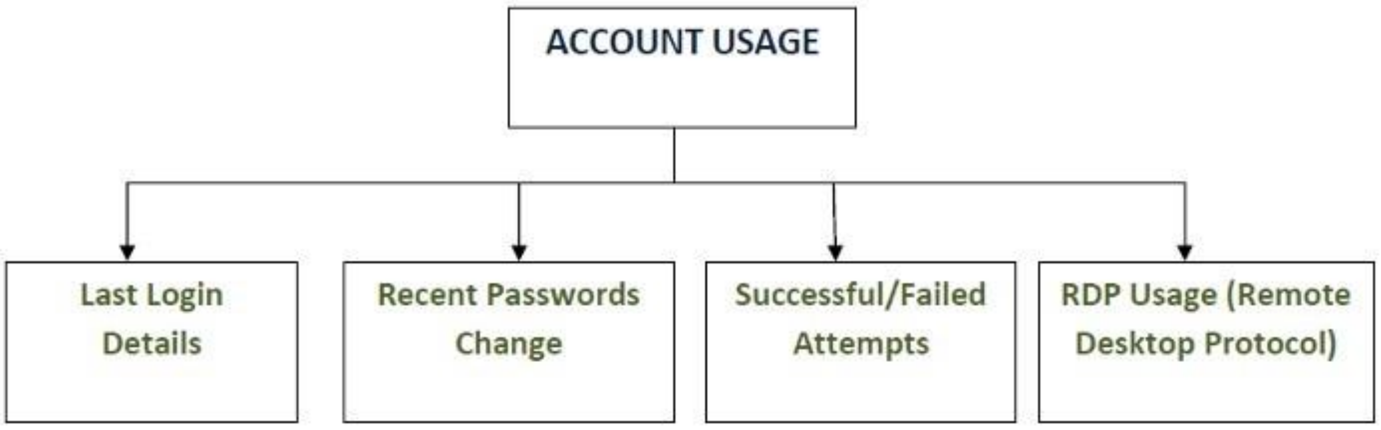

ACCOUNT USAGE
Last Login Details
Recent Passwords Change
Successful/Failed Attempts
RDP Usage (Remote Desktop Protocol)

Figure 11

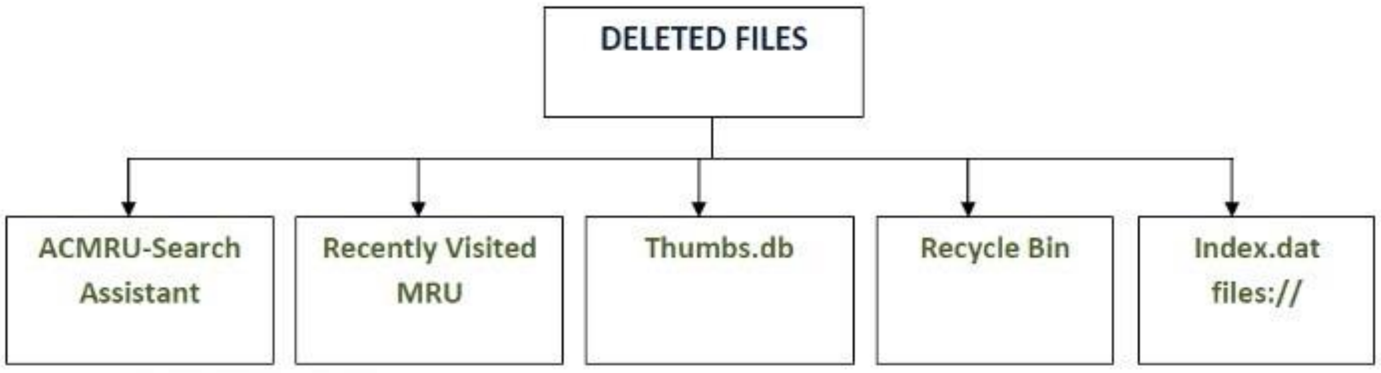


Figure 12

3.2.3 Digital Forensics with Recycle Bin

Microsoft has significantly changed how files and their corresponding details are represented within the Recycle Bin in Windows 7 and Vista. In Windows XP, when files were placed into the Recycle Bin they were placed within a hidden directory named **\Recycler\%SID% where %SID%** is the SID of the user that performed the deletion. The files were renamed **D%drive_letter%%index_number%. %file_extension% where %drive_letter%** is the original drive letter of the file, **%index_number% is an index number, and %file_extension%** is the original file's extension. Additionally, a file named INFO2 was placed in the user's Recycler directory and it contained entries, identified by index number, which described the original file's size and full path/name. In Windows 7 and Vista, Microsoft did away with the INFO2 file and completely changed the way files were named and indexed within the Recycle Bin.

Firstly, the new Recycle Bin is located in a hidden directory named **\$Recycle.Bin\%SID%**, where **%SID%** is the SID of the user that performed the deletion. Secondly, when files are moved into the Recycle Bin, the original file is renamed to $R followed by a set of random characters, but maintaining the original file extension. At the same time a new file beginning with $I followed by the same set of random characters given to the $R file and the same extension, is created; this file contains the original filename/path, original file size, and the date and time that the file was moved to the Recycle Bin. You'll also notice that all of the $I files are exactly 544 bytes long. The behavior is a bit different when you move a directory to the Recycle Bin. The directory name itself is renamed to $R followed by a set of random characters, but the files/directories under that directory maintain their original names. A $I file is created just as when deleting an individual file that contains the original directory name, date/time deleted, and size. When utilizing the information contained in the $I file for forensic purposes, you can safely report that all files found under the $R directory structure within the Recycle Bin were deleted at the same time (and all at once).If a file was previously deleted out of the now deleted directory (but not yet removed from the Recycle Bin), it would have its own $R and $I files and not be grouped with the files that were deleted as part of the directory deletion action. Unfortunately, unlike the INFO2 file, the new $I files are not in plain/readable text. In order to decode a $I files, you must use a forensic tool that has the ability to interpret these files or simply open the file up in a hex editor. The file is structured as follows: [11]

- Bytes 0-7: $I File header – always set to 01 followed by seven sets of 00.
- Bytes 8-15: Original file size – stored in hex, in little-endian.
- Bytes 16-23: Deleted date/time stamp – represented in number of seconds since Midnight. Use a program such as Decode to assist with figuring out the exact date/time, if you don't want to do the math
- Bytes 24-543: Original file path/name.

3.2.4 Physical Memory Image in Windows XP

To analyze forensic evidence from physical memory, one must be able to extract the contents of RAM. Ideally, when a process reads from or writes to a location in its virtual address space, that address is available in physical memory for immediate use. But no system is capable of providing enough physical addresses in main memory to map every address of each process' virtual address space. So, when virtual memory space exceeds physical memory space, non-essential data mapped to physical memory is transferred to the system hard disk, to free physical addresses for remapping. This process is known as paging out.

The data is paged to a file on the system hard disk called a swap file. On Windows systems, the swap file is named pagefile.sys and is located in the root directory of the C drive (C:\). The swap file is conceptually just an extension of physical memory to meet the capacity of virtual memory. For ease of management, main memory is divided into fixed sized chunks called pages. While a system is powered on, pages in memory are swapped to and from the swap file as new processes are created and as the existing processes run, as needed to meet the demand of the running processes. Because of the constant swapping of pages in physical memory, the total address space of virtual memory is rarely contained entirely within physical memory, nor is it ever constant. When a forensics imaging program is used to copy the contents of physical memory to a file, that same memory will be altered by the operating system which must create data structures to manage the newly created imaging process.

Furthermore, the imaging program itself will alter physical memory as it copies the data in memory to a file which typically requires a large number of data transfers between various memory resident input/output buffers. Besides affecting memory, it is possible that the program could be receiving false data from a corrupted operating system since a memory imaging program can only access physical memory through the operating system. Thus, with software tools alone, it is not possible to obtain a snapshot of physical memory unaffected by the tool used to measure it. Simply stated, the act of capturing the state of memory causes changes to the state of memory. However, it is possible to obtain a snapshot of physical memory without altering its state by using a bus mastering hardware 6 device connected to the system's memory bus. The device must be connected to the system prior to intrusion, when the system is powered down, but because it communicates with physical memory through the host controller and not the operating system, the problems discussed with software acquisition of physical memory are eliminated.

For those who do not have a bus mastering device to capture memory or did not have the foresight to install said device before an intrusion, snapshots of the unaltered physical memory image are not possible. An undisturbed snapshot is not necessary for forensic analysis because, as will be shown in this document, useful forensic evidence can be extracted from a physical memory image captured with the software tool.

3.2.5 Windows VISTA Forensics

As with Windows XP, Vista continues to use NTFS for its file system. The WinFS file system, at one time rumored to be a part of Vista, has not been implemented. While the directory structure utilized by Vista is similar to that of XP, a number of folders typically reviewed by examiners are now in different locations. Vista makes use of reparse points1 to point legacy folders (such as \Documents and Settings) to Vista new file locations. Vista recycle bin functionality differs drastically from that of XP.

The \Recycled or \Recycler folders have been replaced with \$Recycle.Bin at the root of the volume. The INFO2 file, which XP used to track files moving in and out of the recycle bin, is no longer used. In its place are pairs of files. When a file is moved to the recycle bin, it is renamed with a random file name starting with $R, with its extension unchanged from the original deleted file. Accompanying this file is an administrative file with the same random file name and extension, starting with $I. This file contains the information which Vista uses to store the deleted files original name and location. Vista provides the ability to view thumbnails in four sizes. A separate thumbs database is created for each of these sizes and stored in a user's folder: \User\\AppData\Local\Microsoft\Windows\Explorer The files are no longer stored in the directories containing the files viewed, as they were in XP.

3.2.6 Digital Forensics-Shimcache

The Windows Registry is an important component of the OS and applications functionality, maintains many aspects of its configuration and plays a key role on its performance. The Windows Registry is the heart and soul of modern Windows operating systems. In any case, from a forensics perspective, the Windows registry is a treasure trove of valuable artifacts. Among these artifacts you might be looking at System and Configuration Registry Keys, Common Auto-Run Registry Keys, User Hive Registry keys or the Application Compatibility Cache a.k.a. ShimCache. In this article we will look into the Application Compatibility Cache. ShimCache.

When performing Live Response or dead box forensics on Windows operating systems one of the many artifacts that might be of interest when determining which files have been executed or accessed is the ShimCache. In our last article we mentioned the Prefetch where you could get evidence about a specific file being executed on the system. However, on Windows Servers operating systems, the Prefetch is disabled by default. This means the ShimCache is a great alternative and also a valuable source of evidence. Let's start with some background about the ShimCache. Microsoft introduced the ShimCache in Windows 95 and today it remains a mechanism to ensure backward compatibility of older binaries into new versions of Microsoft operating systems. When new Microsoft operating systems are released, some old and legacy application might break. To fix this Microsoft has the ShimCache which acts as a proxy layer between the old application and the new operating system.

The interesting part is that from a forensics perspective the ShimCache is valuable because the cache tracks metadata for binary that was executed and stores it in the ShimCache. Through ShimCache we can obtain information about all binaries that have been executed in the system since it was rebooted and track its size and last modified date. In addition, the ShimCache tracks executables that have not been executed but were browsed, for example through explorer.exe. This makes a valuable source of evidence, such as, to track executables that were on the system but weren't executed or consider an attacker that used a directory on a system to move around his toolkit. The ShimCache either directly from memory or by querying the registry after system shutdown we can, in this case, confirm the evidence found in the Prefetch artifacts.

On a Windows Server system because by default the Prefetch is disabled the ShimCache becomes a more valuable artifact. Given the availability of this artifact across all Windows operating systems, the information obtained from the ShimCache can be valuable to an investigation. In this case, the ShimCache supported the findings of Prefetch on **regedit.exe and rundll32.exe** being executed on the system. Essentially, this file is maintained in **%SYSTEMROOT%\AppCompat\Programs\** directory and keeps metadata (PATH and filename) about executables that are new in the system since the last time the service Application Experience was run. On Windows 8 this file has been replaced with a registry HIVE called **amcache.hve** which contains more metadata.

From this file you can retrieve every executable that ran on the system, the PATH, last modification time & time created, SHA1 and PE properties. Meanwhile, on Windows 7 you can also have the amcache.hve if you have installed KB2952664. However, the ShimCache has not only been used from a defensive perspective. From an offensive perspective, the ShimCache has been used several times by attackers as well.

3.2.7 A Forensic Comparison: Windows 7 and Windows 8

In this section, we examine the forensic similarities and differences for Windows 7 and Windows 8.

File Creation and Deletion Artifacts

Similarities

Both in Windows 7 and Windows 8 you are able to recover deleted files even after the recycle bin has been emptied. Using the recover folders task within the evidence processor for EnCase, you can uncover files for both operating systems quite easily, though you will need to manually search through the recovered folders and files. The recover folders task searches through the unallocated clusters of the file system as well as the Master File Table to locate previously deleted files and folders. Once the evidence processor completes sifting through the separate evidence files for Windows 7 and Windows 8, you can search the recovered file folders for the files that had been deleted during the data generation process. You simply have to open the Recovered Folders virtual folder within the root of the partition of the forensic image.

Differences

There are essentially no significant differences in regards to file creation/deletion artifacts between Windows 7 and Windows 8. It may take longer to find the previously created and then deleted files for Windows 8 than it does for Windows 7, but it is still possible to find them through the same process, so this shouldn't be considered a difference. Ultimately, you are able to use EnCase and FTK to recover files that you created as well as files that have been deleted for both Windows 7 and Windows 8.

Windows 8 Registry Artifacts

No discussion on Windows 8 forensic artifacts would be complete without mentioning the changes within the Windows registry. Forensic investigators should be familiar with the standard function of the Windows Registry, which is a central hierarchical database used to store information necessary to configure the system for one or more users, applications, and hardware devices. The registry is considered the heart and soul of the Windows operating system, containing a massive amount of data of forensic significance. The mining of such data can be an unnerving task due to its size and complexity. Fortunately for forensic investigators, Microsoft altered the Windows 8 registry only slightly from its predecessor Windows 7. Nonetheless, the new operating system brought on new registry changes and artifacts.

SAM Registry Artifacts

The Security Accounts Manager (SAM) file is present in the same manner as it was in previous versions of Windows operating systems. The Windows 8 SAM stores users' passwords in a hashed format (in NTLM hash) for both local and Microsoft login accounts. The SAM key stores user names that are used for login and the user’s RID (Relative Identifier) for each account. The addition of the immersive user interface (UI) also brought on new artifacts such as the internet user name and user’s tile registry key. These keys can be found within the following locations within the registry:

Internet User Name
%SystemRoot%\Windows\System32\Config\SAM\
Domains\Account\Users\Internet User Name

User's Tile
%SystemRoot%\Windows\System32\Config\SAM\
Domains\Account\Users\UserTile

SOFTWARE Registry Artifacts

The software key contains information about the operating system, such as the version, when it was installed, the registered owner, the last user to log on, and the members of a created user group. With the addition of metro apps in Windows 8, new registry keys were added. These include a registry key that shows what metro apps were installed on the system and what user account installed such metro apps. These keys can be found within the following locations within the SOFTWARE registry hive.

Metro Apps Installed on System
Microsoft\Windows\CurrentVersion\Appx\AppxAllUser
Store\Applications

User Account Installed Metro Apps
Microsoft\Windows\CurrentVersion\Appx\AppxAllUser
Store\%SID%

NTUSER.DAT Registry Artifacts

The NTUSER.DAT is a registry entry that stores information that is specific to the Windows user. If there are multiple user accounts on a particular operating system, there are also multiple NTUSER.DAT files; one created for each individual user. NTUSER.DAT stores data that is specific to a particular user, such as which files they opened, which applications they used, and Web sites they visited. The structure and usage of the NTUSER.DAT files have not changed in Windows 8, with many of the same residual forensic artifacts as were present in Windows 7.

There is, however, one registry entry that is of significance to forensic investigators, new to Windows 8, which is that of the "TYPEDURLsTIME" entry. This entry is stored in binary form and denotes the number of 100-nanosecond intervals since January 1, 1601 at 00:00:00 GMT. The FILETIME structure consists of two 32-bit values that combine to form a single Little Endian 64-bit value that can be correlated to URLs found in the TypedURLs based upon the corresponding number sequence that the URL was typed into the browser. This information can be found in the following location:

TypedURLsTime
%SystemRoot%\Users\%User%\NTUSER.DAT\Software\
Microsoft\Microsoft\InternetExplorer\TypedURLsTime

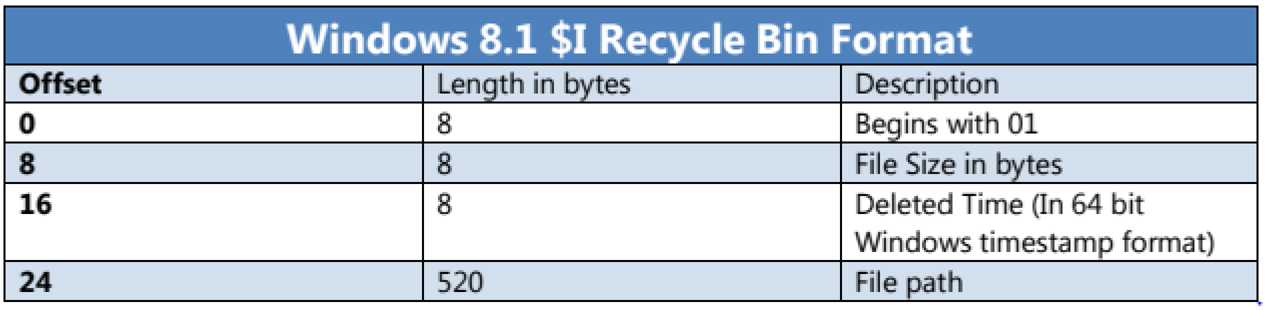
The pre-releases of Windows 8 gave a look into the future of operating system forensics and what difficulties may lie ahead for investigators on the horizon. Although the under-the-hood structure of the operating system was not altered too drastically, Windows 8 brings new challenges to the forensic examination that were not present before. Windows 8 is more interconnected than previous versions, fully utilizing social media and internet utilities such as roaming accounts and cloud storage. This could pose complications in certain investigations as to the legality of accessing such "cloud-based" data. This article delves into artifacts of the "Release Preview" which was unveiled in late May 2012, nearly five months prior to its official release in order to identify and investigate forensically relevant changes to the new operating system.

It is assumed that some of these artifacts will be slightly different upon final release of the operating system and new artifacts will be added. In addition, there was a multitude of specific forensic artifacts that were not addressed in this article. For this reason, it is important that continued research be conducted well beyond the first public release of the operating system this fall.

3.2.8 A Forensic Comparison: Windows 8 and Windows 10

Recycle Bin Analysis Windows 8

- The $I format contains metadata including the file size, deleted time and the file path.
- The $R file contains the deleted file itself.
- The $I file is formatted in the following manner in Windows 8.1



Windows 8.1 $I Recycle Bin Format

Offset	Length in bytes	Description
0	8	Begins with 01
8	8	File Size in bytes
16	8	Deleted Time (In 64 bit Windows timestamp format)
24	520	File path

Figure 13

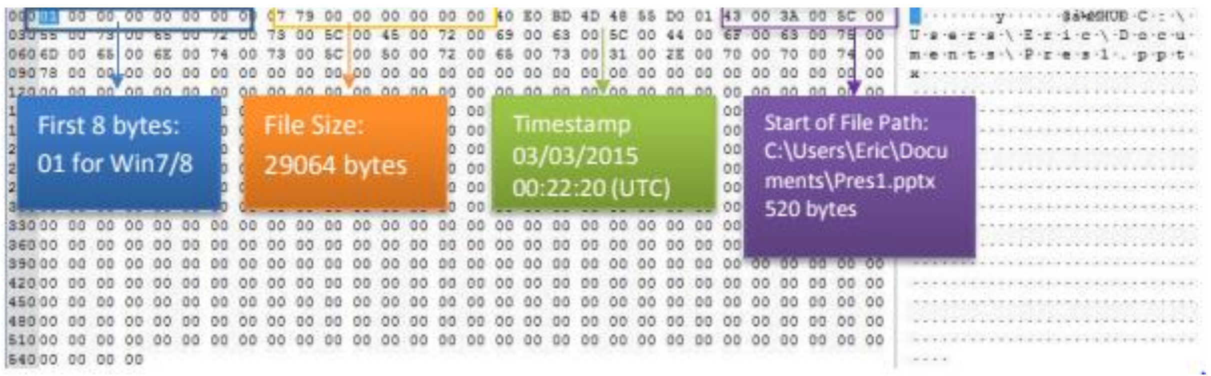
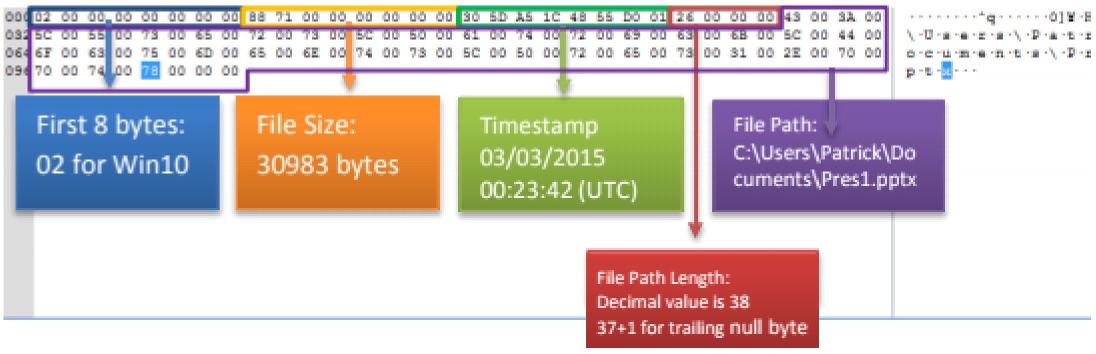


Figure 14

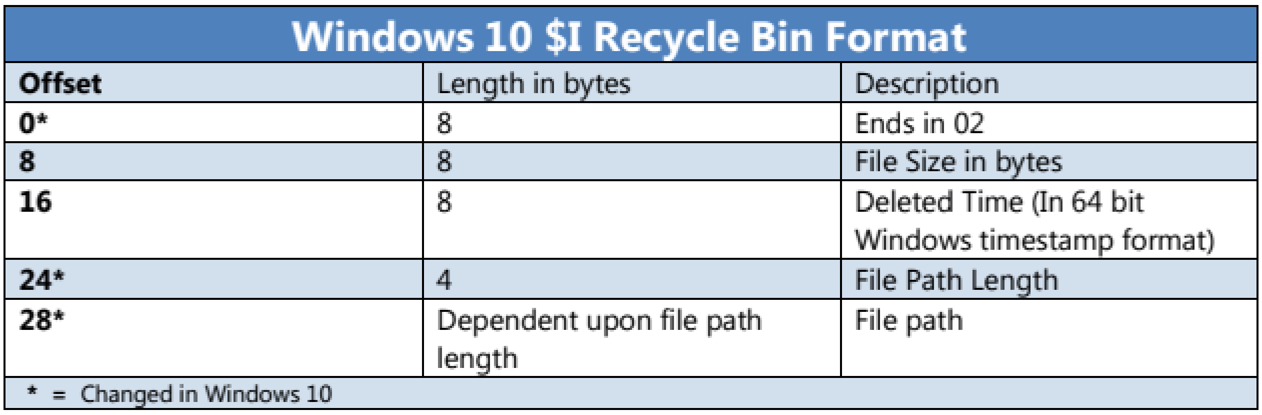
Recycle Bin Analysis Windows 10

Below is a screenshot of a $I file in Windows 10. The first offset value is 8 bytes long, but it starts with a value of 02. Then, we see the 8 bytes related to the file size, followed by the deleted time matching the data generation sheet. The 4byte value at offset 24, is the file path length for the deleted file. You take the byte value and convert it to decimal using Little Endian and add 1 for the trailing null byte. The rest of the file is no longer 520 bytes and is instead based off the file name, as seen below. It appears that the end of this file is marked by three bytes of continuous zeros.



```
000 02 00 00 00 00 00 00 00 88 71 00 00 00 00 00 00 30 5D A5 1C 48 55 D0 01 26 00 00 00 43 00 3A 00
032 5C 00 55 00 73 00 65 00 72 00 73 00 5C 00 50 00 61 00 74 00 72 00 69 00 63 00 6B 00 5C 00 44 00
064 6F 00 63 00 75 00 6D 00 65 00 6E 00 74 00 73 00 5C 00 50 00 72 00 65 00 73 00 31 00 2E 00 70 00
096 70 00 74 00 78 00 00 00
```

Figure 15



Windows 10 $I Recycle Bin Format		
Offset	Length in bytes	Description
0*	8	Ends in 02
8	8	File Size in bytes
16	8	Deleted Time (In 64 bit Windows timestamp format)
24*	4	File Path Length
28*	Dependent upon file path length	File path
* = Changed in Windows 10		

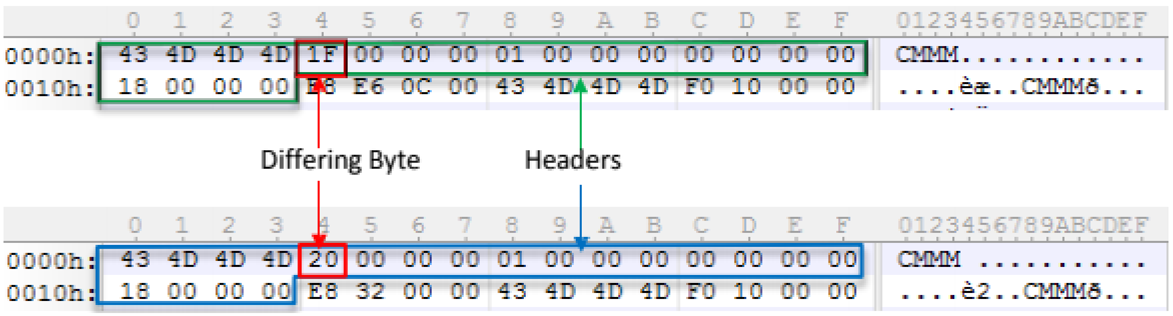
Figure 16

3.2.9 Thumbnails

Thumbnail artifacts can be important to investigators when dealing with potential evidence found in images. In some versions of Windows, thumbnail data is maintained even when the image itself is deleted. Windows XP had a **thumbs.db** file that stored the thumbnail image of every file untilWindows 7 removed this functionality and replaced the thumbs.db folder with a thumbcache.db file located in:

C:\Users\<USERNAME>\AppData\Local\Microsoft\Windows\Explorer

Thumbnail cache header Windows 8



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 43 4D 4D 4D 1F 00 00 00 01 00 00 00 00 00 00 00  CMMM............
0010h: 18 00 00 00 E8 E6 0C 00 43 4D 4D 4D F0 10 00 00  ....èæ..CMMMð...
```

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 43 4D 4D 4D 20 00 00 00 01 00 00 00 00 00 00 00  CMMM ...........
0010h: 18 00 00 00 E8 32 00 00 43 4D 4D 4D F0 10 00 00  ....è2..CMMMð...
```

Thumbnail cache header Windows 10

Figure 17

Windows 10 removes the thumbs.db file once again, storing the thumbnails in the same location as Windows 7:

C:\Users\<USERNAME>\AppData\Local\Microsoft\Windows\Explorer.

The file header for Windows 10's thumbnail cache is only slightly different from 8.1, with one very simple change. Windows 10 has the value of 0x20 instead of 0x1F at offset 4. When one converts these two hex values to decimal figures, they are 31 and 32 respectively. Although a minor alteration, it affects the tools that help investigators view thumbnail caches, and these tools will need to be updated to become compatible with Windows 10. One thing to note is that the **thumbnails.db** files were present in Windows10, so it’s possible that this behavior may change before release.

OneDrive

OneDrive log files are stored in two separate but similar locations on Windows 10. Within this folder there are four types of files: SyncEngine.odl, TraceCurrent.ETL, TraceArchive.ETL, and SyncDiagnostics.txt.

SyncEngine.odl

The SyncEngine file in this directory is the most common file found. The odl file extension is most often used in C++ applications and references many .cpp files such as **filetransferwatcher.cpp** and **localchanges.cpp**. Each file is created with a timestamp and is exactly 1,025 KB. These files appear to be logs of operations that have been performed, but because of the .cpp file references it’s possible that they are used for the actual function of OneDrive syncing. When a file is synced to OneDrive, a SyncEngine file is created and the file will sync filenames and file hashes among the other logs. It’s possible that OneDrive is submitting these hashes to the OneDrive servers to verify file integrity.

Functional Differences

Within these folders, both operating systems have the same file types; the only difference is the way that OneDrive functions on Windows 10. On Windows 8, Windows utilizes “smart folders,” a directory that is viewable to the user, but the contents of these directories aren’t actually on the computer. The user has to manually take each directory offline in order to store each file permanently offline otherwise the OneDrive data is only downloaded when needed. What this means for the end user is that he or she could have a computer with only 256 GB of storage and still be able to browse a directory of OneDrive folders that contained 1 TB of data. This also indicates that logs for OneDrive would reference folders that weren’t physically on the computer OneDrive in Windows 10 differs from Windows 8.1 in that “smart folders” are no longer supported.

When the user launches OneDrive, they are asked which folders they would like to sync offline, and the only way to view other OneDrive folders is to add more folders. In terms of OneDrive logs, it appears that folders which aren’t synced all the time are stored in the OneDrive TraceArchive and TraceCurrent

Trace. ETL

TraceArchive.ETL and TraceCurrent.ETL are logging files which appear to contain the remnants of the smart folder feature in Windows 8. Unfortunately, while event viewer can open them, it doesn’t produce any useful or readable information. However, analyzing the file in notepad seems to work for rudimentary forensics.

Forensic Analysis of Windows 7 Jump Lists

The release of Microsoft Windows 7 introduced a new feature known as Jump Lists which present the user with links to recently accessed files grouped on a per-application basis. The records maintained by the feature have the potential to provide the forensic computing examiner with a rich source of evidence during examinations of computers running the Microsoft Windows 7 Operating System. The Jump List feature supplies the user with a graphical interface associated with each installed application which lists files that have been previously accessed by that application. The feature is enabled as standard and the default setting is to show the 10 most recently accessed files per application, although it is possible to adjust that figure to a maximum of 60.

Windows 10 Jump List Forensics

When Microsoft released Windows 7, a new artifact was released to the forensic world, Jump Lists. Since that time most examiners have become used to examining this artifact and reporting on the results. Jump Lists are potentially a valuable source of evidence that can point directly to a user’s interactions with the computer. Although Jump Lists are a function of the operating system, the service itself can be configured by the user. To enhance the user experience in Windows 10, the creation of jump lists and the recording of opened items is on by default. Users can elect to turn off the service. Unlike previous versions of Windows, a user cannot easily change the number of items displayed in each Jump List from the default 10. Rather, users are left with a simple on/off switch. One point that should be clear when analyzing a Windows computer is that Jump Lists are indicative of user activity. Essentially Jump Lists track files accessed by a user, therefore they will assist in most examinations where a user’s actions on the computer are the focus of the analysis. The actual Jump List files are created on a per user basis, and located: **Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\.**

- **AUTOMATICDESTINATIONS-MS**

\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations-ms are created automatically when a user interacts with the system performing such functions as opening applications or accessing files. The actual Jump List items are contained within OLE containers that are essentially named for the application for which the relevant file was accessed. The application ID is normally set by either the application, or the OS when the application is run.

- **CUSTOMDESTINATIONS -MS**

under*\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations-ms* are created when a user “pins” a file to the Start Menu or Task Bar.

Jump Lists are one of the most important forensic artifacts of recent times. Like many forensic artifacts, the intent of Jump Lists is to provide users with increased usability and convenience. However, examiners can take advantage of this service and gather much critical insight into the user’s computer habits, knowledge and activities.

Chapter 4 - Conclusion

Security is an important field that is increasingly gaining attention as the internet expands. Computer security attempts to ensure the confidentiality, integrity, and availability of computing systems and their components. Three principal parts of a computing system are subject to attacks: hardware, software, and data. These three, and the communications among them, are susceptible to computer security vulnerabilities. In turn, those people and systems interested in compromising a computer can devise attacks that exploit said vulnerabilities. Security situations arise in many everyday activities, although sometimes it can be difficult to distinguish between a security attack and an ordinary human or technological breakdown. Alas, clever attackers realize this confusion, so they may construct their attack in such a way as to make it appear like a simple, random failure.

A threat is an incident that could cause harm. A vulnerability is a weakness through which harm could occur. These two problems combine: Either without the other causes no harm, but a threat exercising a vulnerability means damage. To control such a situation, we can either block or diminish the threat, or close the vulnerability (or both). Seldom can we achieve perfect security: no viable threats and no exercisable vulnerabilities. Sometimes we fail to recognize a threat, or other times we may be unable or unwilling to close a vulnerability. Incomplete security is not a bad situation; rather, it demonstrates a balancing act: Control certain threats and vulnerabilities, apply countermeasures that are reasonable, and accept the risk of harm from encountered cases.

An attacker needs three things: method—the skill and knowledge to perform a successful attack; opportunity—time and access by which to attack; and motive—a reason to want to attack. Alas, none of these three is in short supply, which means attacks are inevitable. Attackers leverage threats that exploit vulnerabilities against valuable assets to cause harm, and we hope to devise countermeasures to eliminate means, opportunity, and motive. These concepts are the basis we need to study, understand, and master computer security. Countermeasures and controls can be applied to the data, the programs, the system, the physical devices, the communications links, the environment, and the personnel. Sometimes several controls are needed to cover a single vulnerability, but sometimes one control addresses many problems at once. With Microsoft constantly updating its security features in each occurring Windows version, as long as users stay updated on current software that protect their machines they will be able to successfully fend of potential attacks.

Moving forward, people will do well to stay informed and upgrade their security protections on Windows as well as access information on their personal devices using protected networks because nowadays nothing is completely safe. The security protection issues on Windows are always one of the main topics for researchers and developers to investigate the appropriate solutions. From the perspective of this thesis, we have attempted to find an optimum and appropriate security solution for the specific version of Windows at hand. There is a scope for proposing improved guidelines to overcome the future challenges like physical security, espionage, transparency, data ownership, hypervisor viruses and malicious insiders on security for Windows.

References

[1] https://insights.sei.cmu.edu/cert/2016/11/windows-10-cannot-protect-insecure-applications-like-emet-can.html

[2] https://www.rsaconference.com/writable/presentations/file_upload/exp-r01_patching-exploits-with-duct-tape-bypassing-mitigations-and-backward-steps.pdf

[3] http://security.cs.rpi.edu/~gaasem/winexp/IndependentStudy.pdf

[4]http://media.blackhat.com/bh-us-12/Briefings/M_Miller/BH_US_12_Miller_Exploit_Mitigation_Slides.pdf

[5] http://www.semantiscope.com/research/BHDC2010/BHDC-2010-Slides-v2.pdf

[6] http://security.stackexchange.com/questions/20497/stack-overflows-defeating-canaries-aslr-dep-nx

[7] http://web.mit.edu/~bdaya/www/Network%20Security.pdf

[8]http://www.hakim.ws/BHUSA08/speakers/Sotirov_Dowd_Bypassing_Memory_Protections/BH_US_08_Sotirov_Dowd_Bypassing_Memory_Protections.pdf

[9] http://resources.infosecinstitute.com/windows-memory-protection-mechanisms/#gref

[10] https://www.blackhat.com/presentations/bh-usa-08/Sotirov_Dowd/bh08-sotirov-dowd.pdf

[11] https://dereknewton.com/2010/06/recycle-bin-forensics-in-windows-7-and-vista/

[12] https://articles.forensicfocus.com/2014/04/14/windows-forensics-and-securit

[13] http://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1974&context=theses

[14] https://countuponsecurity.com/2016/05/18/digital-forensics-shimcache-artifacts/

[15] https://www.forensicmag.com/article/2012/09/microsoft-windows-8-forensic-first-look

[16] http://www.dtic.mil/dtic/tr/fulltext/u2/a457305.pdf

[17] https://www.blackbagtech.com/blog/2017/01/12/windows-10-jump-list-forensics/

[18] https://articles.forensicfocus.com/2012/10/30/forensic-analysis-of-windows-7-jump-lists/

[19] Patrick Leahy Center for Digital Investigation (LCDI)

[20] https://arstechnica.com/information-technology/2013/10/windows-8-1-includes-seamless-automatic-disk-encryption-if-your-pc-supports-it/