Reverse Engineering and Malware Analysis

Chapter-1

Introduction to

Malware and Reverse

Engineering

Malware

- Malware, short for malicious software, refers to any intrusive software developed by cybercriminals (often called hackers) to steal data and damage or destroy computers and computer systems.
- Malware is a catch-all term for any type of malicious software designed to harm or exploit any programmable device, service or network. Cybercriminals typically use it to extract data that they can leverage over victims for financial gain. That data can range from financial data, to healthcare records, to personal emails and passwords—the possibilities of what sort of information can be compromised have become endless.
- Malware—short for "malicious software"—is any software code or computer program intentionally written to harm a computer system or its users. Almost every modern cyberattack involves some type of malware. These malicious programs can take many forms, ranging from highly damaging and costly ransomware to merely annoying adware, depending on what cybercriminals aim to do.

What is the intent of malware?

- The goal of malware is to cause havoc and steal information or resources for monetary gain or sheer sabotage intent.
- **Intelligence and intrusion:** Exfiltrates data such as emails, plans, and especially sensitive information like passwords.
- **Disruption and extortion:** Locks up networks and PCs, making them unusable. If it holds your computer hostage for financial gain, it's called ransomware.
- **Destruction or vandalism:** Destroys computer systems to damage your network infrastructure.
- **Steal computer resources:** Uses your computing power to run botnets, cryptomining programs (cryptojacking), or send spam emails.
- **Monetary gain:** Sells your organization's intellectual property on the dark web.

Why do cybercriminals use malware?

- Malware encompasses all types of malicious software, including viruses, and cybercriminals use it for many reasons, such as:
- Tricking a victim into providing personal data for identity theft
- Stealing consumer credit card data or other financial data
- Assuming control of multiple computers to launch denial-of-service attacks against other networks
- Infecting computers and using them to mine bitcoin or other cryptocurrencies

Types of Malware

- **Virus**
- Viruses are a subgroup of malware. A virus is malicious software attached to a document or file that supports macros to execute its code and spread from host to host. Once downloaded, the virus will lie dormant until the file is opened and in use.
- A virus usually comes as an attachment in an email that holds a virus payload, or the part of the malware that performs the malicious action. Once the victim opens the file, the device is infected.
- **Worms**
- A worm is a type of malicious software that rapidly replicates and spreads to any device within the network. Unlike viruses, worms do not need host programs to disseminate. A worm infects a device through a downloaded file or a network connection before it multiplies and disperses at an exponential rate. Like viruses, worms can severely disrupt the operations of a device and cause data loss.
- Worms have the ability to copy themselves from machine to machine, usually by exploiting some sort of security weakness in a software or operating system and don’t require user interaction to function. Worms often attack a computer’s memory or hard drive

Types of Malware

- **Ransomware**
- One of the most profitable, and therefore one of the most popular, types of malware amongst cybercriminals is ransomware.
- Ransomware is malicious software that gains access to sensitive information within a system, encrypts that information so that the user cannot access it, and then demands a financial payout for the data to be released. Ransomware is commonly part of a phishing scam. By clicking a disguised link, the user downloads the ransomware.
- **Trojan**
- Trojan viruses are disguised as helpful software programs. But once the user downloads it, the Trojan virus can gain access to sensitive data and then modify, block, or delete the data. This can be extremely harmful to the performance of the device. Unlike normal viruses and worms, Trojan viruses are not designed to self-replicate.
- Trojans masquerade as harmless applications, tricking users into downloading and using them. Once up and running, they then can steal personal data, crash a device, spy on activities or even launch an attack.

Types of Malware

- **Spyware**
- Spyware is malicious software that runs secretly on a computer and reports back to a remote user. Rather than simply disrupting a device's operations, spyware targets sensitive information and can grant remote access to predators. Spyware is often used to steal financial or personal information. A specific type of spyware is a keylogger, which records your keystrokes to reveal passwords and personal information.
- **Adware**
- Adware is malicious software used to collect data on your computer usage and provide appropriate advertisements to you. While adware is not always dangerous, in some cases adware can cause issues for your system. Adware can redirect your browser to unsafe sites, and it can even contain Trojan horses and spyware.
- Adware programs push unwanted advertisements at users and typically display blinking advertisements or pop-up windows when you perform a certain action. Adware programs are often installed in exchange for another service, such as the right to use a program without paying for it.

Types of Malware

- **Fileless malware**
- Fileless malware is a type of memory-resident malware. As the term suggests, it is malware that operates from a victim's computer's memory, not from files on the hard drive. Because there are no files to scan, it is harder to detect than traditional malware. It also makes forensics more difficult because the malware disappears when the victim computer is rebooted.
- It uses legitimate programs to infect a computer. Fileless malware registry attacks leave no malware files to scan and no malicious processes to detect **OR** is a kind of attack that uses vulnerabilities in legitimate software programs like web browsers and word processors to inject malicious code directly into a computer's memory.
- Many fileless malware attacks use PowerShell, a command line interface and scripting tool built into Microsoft Windows operating systems. Hackers can execute PowerShell scripts to change configurations, steal passwords, or do other damage.
- **Ex:** Malicious macros are another common vector for fileless attacks. Apps like Microsoft Word and Excel allow users to define macros, sets of commands that automate simple tasks like formatting text or performing calculations. Hackers can store malicious scripts in these macros; when a user opens the file, those scripts automatically execute

Other Types of Malwares

- **Bots/Botnets**
- A bot is a software application that performs automated tasks on command. They're used for legitimate purposes, such as indexing search engines.
- A botnet is a network of internet-connected, malware-infected devices under a hacker's control. Botnets can include PCs, mobile devices, Internet of Things (IoT) devices, and more.
- **EX:** Hackers often use botnets to launch DDoS attacks, which bombard a target network with so much traffic that it slows to a crawl or shuts down completely.
- **Rootkit/ Remote access malware**
- Rootkits were not originally designed as malware, but they have become a common attack vector for threat actors.
- A rootkit is software that gives malicious actors remote control of a victim's computer with full administrative privileges. Hackers can then use these elevated permissions to do virtually anything they want, like adding and removing users or reconfiguring apps. Hackers often use rootkits to hide malicious processes or disable security software that might catch them.

Other Types of Malwares

- **Scareware**
- Cybercriminals scare us into thinking that our computers or smartphones have become infected to convince victims to purchase a fake application. In a typical scareware scam, you might see an alarming message while browsing the Web that says “Warning: Your computer is infected!” or “You have a virus!” Cybercriminals use these programs and unethical advertising practices to frighten users into purchasing rogue applications.
- **Cryptojackers**
- A cryptojacker is malware that takes control of a device and uses it to mine cryptocurrency, like bitcoin, without the owner's knowledge. Essentially, cryptojackers create cryptomining botnets. Mining cryptocurrency is an extremely compute-intensive and expensive task. Cybercriminals profit while users of infected computers experience performance slowdowns and crashes.
- **Ex:** Cryptojackers often target enterprise cloud infrastructure, allowing them to marshal more resources for cryptomining than targeting individual computers.

Other Types of Malwares

- **Mobile Malware**
- As the name suggests, mobile malware is designed specifically to target mobile devices. This kind of malware has become more common not just with the proliferation of smart phones, but with the increase of mobile and tablet use by organizations and employees.
- Mobile malware can employ several tactics, including spying and recording texts and phone calls (a form of spyware), impersonating common apps, stealing credentials, or accessing data on the device. Mobile malware often spreads through **smishing, also known as SMS phishing**, which is a form of phishing that comes through text messages.
- Other forms of mobile malware include remote access tools, bank Trojans, and crypto mining malware. As phones become a more valuable tool in the workplace, often including the device used in MFA applications, they become a larger target for threat actors.
- **Wiper Malware**
- A wiper is a type of malware with a single purpose: to erase user data and ensure it can’t be recovered. Wipers are used to take down computer networks in public or private companies across various sectors. Threat actors also use wipers to cover up traces left after an intrusion, weakening their victim’s ability to respond.

Analysis of Malware

- Malware analysis is the process of understanding the behavior and purpose of a suspicious file or URL. The output of the analysis aids in the detection and mitigation of the potential threat.
- The **key benefit of malware analysis** is that it helps incident responders and security analysts:
- Pragmatically triage incidents by level of severity
- Uncover hidden indicators of compromise (IOCs) that should be blocked
- Improve the efficacy of IOC alerts and notifications
- Enrich context when threat hunting

Goal of Analysis

- To understand the type of malware and its functionality.
- Determine how the system was infected by malware and define if it was a targeted attack or a phishing attack.
- How malware communicates with attacker.
- Future detection of malware and generating signatures.

Types of Malware Analysis

- **Static Malware Analysis**
- Static malware analysis looks for files that may harm your system without actively running the malware code, making it a safe tool for exposing malicious libraries or packaged files. Static malware analysis can uncover clues regarding the nature of the malware, such as filenames, hashes, IP addresses, domains, and file header data. The malware can be observed using a variety of tools, such as network analyzers.
- **Dynamic Malware Analysis**
- Dynamic malware analysis uses a sandbox, which is a secure, isolated, virtual environment where you can run suspected dangerous code. Security professionals can closely monitor the malware in the sandbox without worrying about infecting the rest of the system or network, allowing them to gather more information about the malware.

Types of Malware Analysis

- **Hybrid Malware Analysis**
- Hybrid malware analysis combines both static and dynamic techniques. For example, if malicious code makes changes to a computer's memory, dynamic analysis can detect that activity. Then, static analysis can determine exactly what changes were made.
- **Code analysis –** It is a process of analyzing/reverse engineering assembly code. It is combination of both static and dynamic analysis.
- **Behavioral analysis –** It is the process of analyzing and monitoring the malware after execution. It involves monitoring the processes, registry entries and network monitoring to determine the workings of the malware.

Steps in Malware Analysis

- **Identification:** Determining the presence of malware and understanding its characteristics.
- **Acquisition:** Obtaining a copy of the malware for analysis, ensuring proper handling and containment.
- **Preliminary Analysis:** Conducting initial assessments to gather basic information about the malware.
- **Static Analysis:** Examining the malware without executing it to extract metadata and understand its structure.
- **Dynamic Analysis:** Executing the malware in a controlled environment to observe its behavior and effects.
- **Code Analysis:** Analyzing the malware's code to understand its functionality, logic, and potential vulnerabilities.
- **Behavioral Analysis:** Monitoring the malware's actions during execution to identify its interactions with the system and network.
- **Reverse Engineering:** Unpacking and decompiling the malware to understand its inner workings and algorithms.
- **Post-Analysis:** Documenting findings, generating reports, and deriving insights for future prevention and detection.

Malware Analysis Use Cases

- **Incident Response**
- For remediation and recovery to be successful, incident response teams must move quickly, and this is where malware analysis is especially useful. By giving incident responders applicable information for ongoing and upcoming incidents, malware analysis enables them to contain and prevent attacks.
- **Malware Research and Detection**
- To best safeguard your organization, identifying malicious code and understanding how it differs from benevolent code is extremely important. For example, by knowing which sites transmit malicious code, you can blacklist websites that propagate threats.
- **Indicator of Compromise (IOC) Extraction**
- With malware analysis, you can extract indicators of compromise (IOCs) to better understand how malware can attack your system. An IOC is data indicating that a system breach or attack has occurred. You can use this data to understand how your system reacts to attacks, making it easier to detect attacks in the future.

Malware Analysis Use Cases

- **Threat Hunting**
- Threat hunters use malware analysis to identify previously unknown cyberthreats. For example, if you set up a honey trap, which is designed to attract malware and confine it to a homeless area of your network, you can study how the malware behaves and potentially discover a new threat. Using malware analysis in this way may reveal threats that can get past your defenses.
- **Threat Alerts and Triage**
- Malware analysis enables IT teams to better understand how threats work and then use this information to react faster. The right malware analysis tool can send you alerts, prioritizing them according to severity. This way, instead of wasting time tracking down false positives, your security team can focus their energies on the threats that really matter.

Tools for Malware Analysis

- Process Hacker
- Fiddler
- Limon
- PeStudio
- Cuckoo Sandbox
- CrowdStrike Falcon Insight
- Ghidra

Assembly Language

- An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware.
- Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.
- An assembly language is a type of programming language that translates high-level languages into machine language.
- It is a necessary bridge between software programs and their underlying hardware platforms.
- Assembly language relies on language syntax, labels, operators, and directives to convert code into usable machine instruction.
- Assembly language may pass through single-pass or multi-pass assemblers, each with specific uses and benefits.

How Assembly Languages Work

- Fundamentally, the most basic instructions executed by a computer are binary codes, consisting of ones and zeros. Those codes are directly translated into the “on” and “off” states of the electricity moving through the computer’s physical circuits.
- no human would be able to construct modern software programs by explicitly programming ones and zeros. Instead, human programmers must rely on various layers of abstraction that can allow themselves to articulate their commands in a format that is more intuitive to humans.
- Specifically, modern programmers issue commands in so-called “high-level languages,” which utilize intuitive syntax such as whole English words and sentences, as well as logical operators such as “and,” “or,” and “else” that are familiar to everyday usage.

How Assembly Languages Work

- Ultimately, however, these high-level commands need to be translated into machine language. Rather than doing so manually, programmers rely on assembly languages whose purpose is to automatically translate between these high-level and low-level languages. The first assembly languages were developed in the 1940s, and though modern programmers and modern natural language processors spend very little time dealing with assembly languages

Components of Assembly Language

- Syntax
- When writing any code in any program language, there is an observable, specific order of rules that must be followed to allow a compiler to execute the code without error. These rules are defined as the syntax, and they contain criteria such as the maximum number of allowable characters, what characters code lines must start with, or what certain symbols "i.e. a semi-colon" means.
- Label
- A label is a symbol that represents the address where an instruction or data is stored. It's purpose is to act as the destination when referenced in a statement. Labels can be used anywhere an address can be used in assembly languages. A symbolic label consists of an identifier followed by a colon, while numeric labels consist of a single digital followed by a colon.

Components of Assembly Language

- Operators
- Also referred to as commands, operators are logical expressions that occur after the label field. In addition, it must be preceded by at least one white-space character. Operators can either be opcode or directive. Opcode correspond directly to machine instructions, and the operation code includes any register name associated with the instruction. Alternatively, directive operation codes are instructions known by the assembler.
- Directive
- Directives are instructions to the assembler that tell what actions must take place during the assembly process. Directives have the importance of declaring or reserving memory for variables; these variables can be recalled later in processes to perform more dynamic functions. Directives are also used to break programs into different sections.

Components of Assembly Language

- Macro
- An assembly language macro is a template shoe format presents a series or pattern of statements. This sequence of assembly language statements might be common to multiple different programs. A macro facility is used to interpret macro definitions, while a macro call is inserted into the source code where "normal" assembly code would have gone instead of the macro set of statements.
- Mnemonic
- A mnemonic is an abbreviation for an operation. A mnemonic is entered into the operation code for each assemble program instruction to specify a shortened "opcode" that represents a larger, complete set of codes. For example, the mnemonic "multiply by two" has a full set of code that carries out the mnemonic.

```
; ------------------------------------------------------------------------
; Writes "Hello, World" to the console using only system calls. Runs on 64-bit Linux only.
; To assemble and run:
;
;     nasm -felf64 hello.asm && ld hello.o && ./a.out
; ------------------------------------------------------------------------

          global    _start

          section   .text
_start:   mov       rax, 1                  ; system call for write
          mov       rdi, 1                  ; file handle 1 is stdout
          mov       rsi, message            ; address of string to output
          mov       rdx, 13                 ; number of bytes
          syscall                           ; invoke operating system to do the write
          mov       rax, 60                 ; system call for exit
          xor       rdi, rdi                ; exit code 0
          syscall                           ; invoke operating system to exit

          section   .data
message:  db        "Hello, World", 10      ; note the newline at the end
```

Review: Assembly Language Example 1

```
   .text
   .align       2
main:
   subu $29, $29, 32
   sw   $31, 20($29)
   sd   $4, 32($29)
   sw   $0, 24($29)
   sw   $0, 28($29)
loop:
   lw   $14, 28($29)
   mul  $15, $14, $14
   lw   $24, 24($29)
   addu $25, $24, $15
   sw   $25, 24($29)
   addu $8, $14, 1
   sw   $8, 28($29)
   ble  $8, 100, loop
   la   $4, str
   lw   $5, 24($29)
   jal  cout
   move $2, $0
   lw   $31, 20($29)
   addu $29, $29, 32
   jr   $31

   .data
   .align 0
str:
   .asciiz "The answer is "
```

Reverse Engineering

- The process of taking a piece of software or hardware and analyzing its functions and information flow so that its functionality and behavior can be understood. Malware is commonly reverse-engineered in cyber defense.
- **What Is Reverse Engineering Malware?**
- Reverse engineering malware is the process of analyzing malware to understand its functionality and purpose. This process can determine how to remove the malware from a system or create defenses against it
- Reverse engineering malware is challenging, as malware is often designed to be difficult to analyze. Typically, a malware reverse engineering program would be necessary to become proficient at it. Threat actors may use obfuscation techniques, encryption, and other tricks to make the programs more complex.

Reverse Engineering

- Reverse-engineering is the act of dismantling an object to see how it works. It is done primarily to analyze and gain knowledge about the way something works but often is used to duplicate or enhance the object. Many things can be reverse-engineered, including software, physical machines, military technology and even biological functions related to how genes work.
- Software reverse-engineering focuses on a program's machine code-- the string of 0s and 1s that are sent to the logic processor. Program language statements are used to turn the machine code back into the original source code.
- the knowledge gained during reverse-engineering can be used to do a security analysis, gain a competitive advantage or simply to teach someone about how something works, reverse-engineering is the process of gaining that knowledge from a finished object.

Goal of RE

- Often the goal of reverse-engineering software or hardware is to find a way to create a similar product more inexpensively or because the original product is no longer available.
- Reverse-engineering in information technology is also used to address compatibility issues and make the hardware or software work with other hardware, software or operating systems that it wasn't originally compatible with.

How does the reverse-engineering process work?

- There are three general steps common to all reverse-engineering efforts. They include:
- **Information extraction.** The object being reverse-engineered is studied, information about its design is extracted and that information is examined to determine how the pieces fit together. In software reverse-engineering, this might require gathering source code and related design documents for study. It may also involve the use of tools, such as a disassembler to break apart the program into its constituent parts.
- **Modeling.** The collected information is abstracted into a conceptual model, with each piece of the model explaining its function in the overall structure. The purpose of this step is to take information specific to the original and abstract it into a general model that can be used to guide the design of new objects or systems. In software reverse-engineering this might take the form of a data flow diagram or a structure chart.
- **Review.** This involves reviewing the model and testing it in various scenarios to ensure it is a realistic abstraction of the original object or system. In software engineering this might take the form of software testing. Once it is tested, the model can be implemented to reengineer the original object.

Under what circumstances is reverse engineering useful or breaking contracts?

- Reverse-engineering a patented product is generally legal under the Defend Trade Secrets Act, but there are situations where its legality is questionable. Patent owners have legal recourse against anyone copying their inventions.
- Reverse-engineering software for the purpose of copying or duplicating a program may constitute a copyright law violation. Some software licenses specifically prohibit reverse-engineering. Other contractual agreements can also limit the use of reverse-engineering to gain access to code, including terms of service or use notices and nondisclosure and other types of developer agreements.
- Technological protection measures (TPM), such as passwords, encryption and access control devices, are often used to control access to software and other digital copyrighted content. Circumventing TPM can raise legal issues.

Under what circumstances is reverse engineering useful or breaking contracts?

- The various laws pertaining to reverse-engineering include the following:
- Patent law
- Copyright and fair use law
- Trade secret law
- anticircumvention provisions of the Digital Millennium Copyright Act
- Electronic Communications Privacy Act
- Any contract law specific to the product in question

When a reverse-engineering challenge is brought to court, the original owner of the object, system or intellectual property must prove that they created the object or own the patent or copyright. The person or organization doing the reverse-engineering must prove they obtained the information in legal ways.

Under what circumstances is reverse engineering useful or breaking contracts?

- **Legal Doctrines Relating to Reverse Engineering**
- Copyright law (17 U.S. Code § 1201 (f))
- Trade secret law
- The anti-circumvention provisions of the DMCA (17 U.S. Code § 1201)
- Contract laws (EULAs, TOS, TOU, and NDA)
- Electronic Communication Privacy Act (ECPA)

Contract laws

- Contract law varies based on the type of software application but most of the software products include EULA conditions of "**no reverse engineering**" clauses. **Therefore, contract law in most cases limits reverse engineering.**
- **End User License Agreement (EULA):** This is a legal contract between a software developer or vendor and the end-user of the software. These agreements are also known as "click-through" agreements that bind customers to a number of strict terms.
- Following are examples of some common EULA clauses that apply to customers' behavior:
  - "Do not criticize this product publicly."
  - "Using this product means you will be monitored."
  - "Do not reverse-engineer this product."
  - "Do not use this product with other vendor's products."
  - "By signing this contract, you also agree to every change in future versions of it. Oh yes, and EULAs are subject to change without notice."
  - "We are not responsible if this product messes up your computer."

Contract laws

- **Terms of Service notice (TOS)**: This is a legal agreement between a service provider and a person who wants to use that service. For example, access to mobile applications or websites. Using this, service providers can deactivate accounts that do not follow the terms of this agreement.
- It is also known as “Terms and conditions” and comprises phrases which are attached to services and/or products. Services that include these terms are web browsers, e-commerce, web search engines, social media, and/or transport services. Terms of service vary based on product and depend on the service provider, so any comment with respect to reverse engineering the product varies accordingly.

Contract laws

- **Terms of Use Notice (TOU):** It is an agreement in which a user must agree to and abide by in order to use a website or service. It is also referred to as “Terms of Service”, “Terms and conditions” and/or “Disclaimer”. Terms of Use vary based on product and depend on the service provider, so any comment with respect to reverse engineering the product varies accordingly.
- **Non-Disclosure Agreement (NDA):** This is an agreement in which parties agree not to disclose secret information. For example, confidential and proprietary information or trade secrets. It is also known as the Confidentiality Agreement (CA), Confidential Disclosure Agreement (CDA), Proprietary Information Agreement (PIA), or Secrecy Agreement (SA). It is commonly signed between two companies which come under partnership in any business.

Why is reverse engineering necessary?

- Reverse-engineering is often used to create replacement parts when the original parts for legacy equipment are no longer available. Reverse-engineering of computer parts is also done to enhance security.
- For example
- Google's Project Zero identified vulnerabilities in microprocessors using reverse-engineering.
- Network security assessments: Companies doing network security assessments also use reverse-engineering as one of their tools. They divide their security group into two teams. One team simulates attacks, and the other team monitors the network and reverse-engineers the other team's attacks. The information gained from these mock attacks is used to strengthen the corporate network

Purpose of Reverse Engineering

- Legacy Parts Replacement
- Parts Service or Repair
- Failure Analysis
- Parts Improvement
- Diagnostics and Problem-Solving

Interoperability

- **Copyright Protection Extended For Interoperability**
- Interoperability has been defined as "the ability of computer programs to exchange information, and of such programs mutually to use the information, which has been exchanged."
- Achieving interoperability through reverse-engineering can infringe the rights of the original owner. One of the main motives for reverse engineering software is to enable new software products to interact with programs that are already available.
- Courts have recognised that reverse-engineering is important for public benefit and to encourage inventors and businesses thereby maintaining healthy competition in the market

Interoperability

- Software programme contains both elements protected and unprotected.
- Ex:
- In *Computer Associates Int'l Incorporated v. Altai Incorporated*, US court interpreted and ruled that copyright law does not extend protection to interfaces necessary to interoperability. However, it was later ruled in the case that reverse-engineering for a legitimate purpose would not lead to infringement.
- When it comes to software, interoperability is an exception. It is legitimate as it allows new software to function. In that case, there is no need for the person to ask for authorization from the author of the software to obtain the information needed for interoperability. It is pertinent to note that the third party should have the right to use a copy of the target software.

Interoperability

- Software reverse engineering involves disassembly (Translation of a program from machine code into a higher-level programming language) or decompilation (produce source code from compiled code) and intermediate copying of the programme which is seen as an infringement of copyright.
- The source code of a software is protected while the object code is protected indirectly.
- This means that in the very first instance by virtue of reverse-engineering, copyright is infringed as both the source code and object code is copyrightable. However, One of the most crucial decision w.r.t to the copyright protection for Reverse engineering is *Sega v. Accolade* in which the US Court was of the view that if the purpose of it is to achieve compatibility, then such usage would be covered under fair use doctrine and would not amount to infringement of copyright.

Interoperability

- In India, the answer to this question is not clear especially because the Copyright Act does not define either Decompilation or Interoperability. Section 52(1)(aa) to (ad) of the Copyright Act provides for exceptions in case of computer programme and software. Specifically, Section 52(1)(ab) and (ac) includes the concept of reverse engineering,
- wherein (ab) talks about doing of an act to obtain necessary information for interoperability where such information is not readily available and (ac) allows observation, study or test of functioning in order to determine the ideas and principles of the elements of the programme. This provision essentially allows for decompiling a computer programme.

- Regular Audits and Inspections: Conduct regular audits and inspections of businesses and entities suspected of patent infringement, particularly in industries vulnerable to reverse engineering.
- Regular Audits and Compliance: Conduct regular audits and compliance checks to ensure that technology transfer agreements are being honored by both parties and that intellectual property rights are being respected.
- Regular Capacity Audits: Conduct regular capacity audits and assessments to identify gaps and areas where further training is needed.