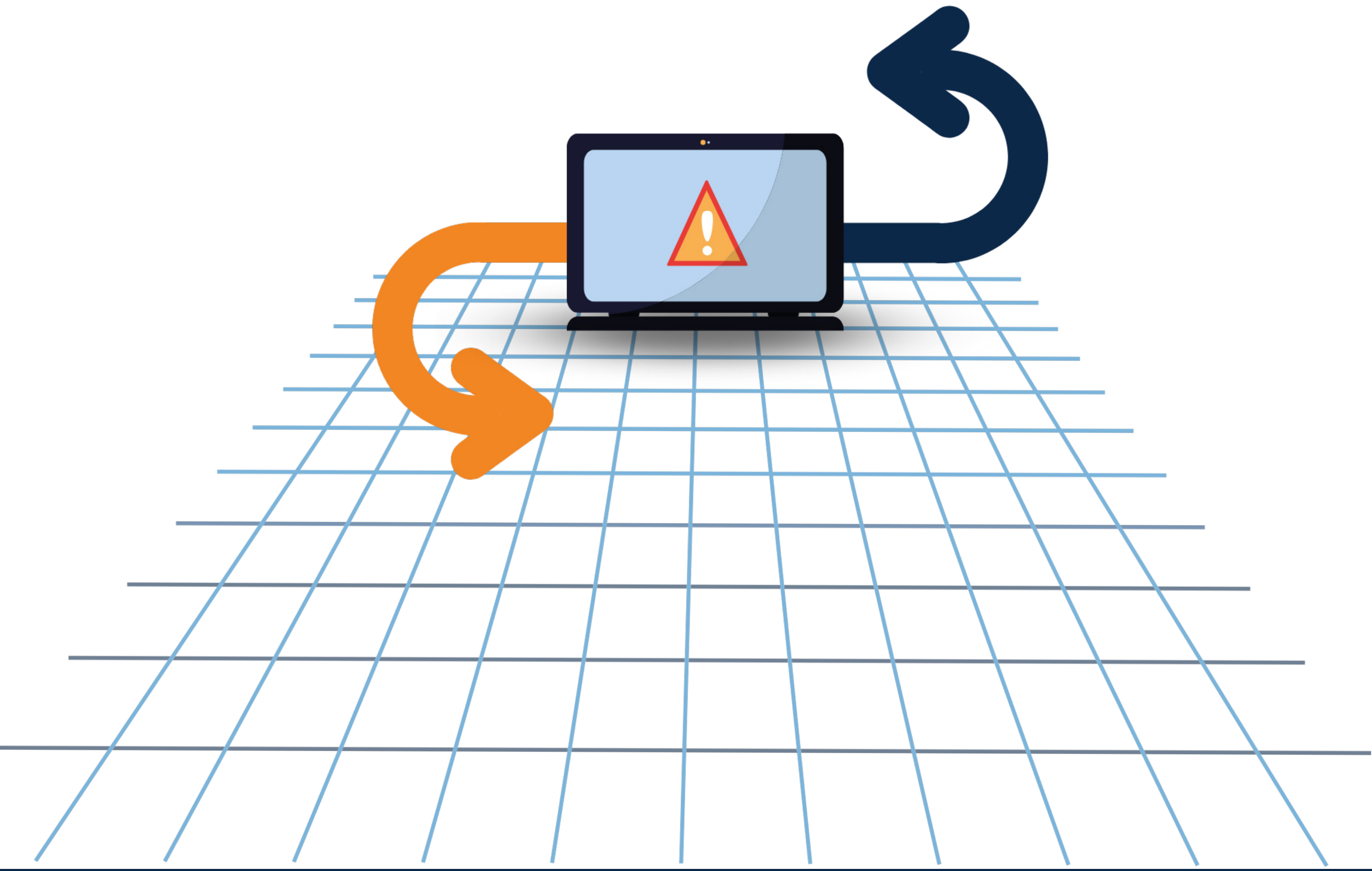
Reverse Engineering:

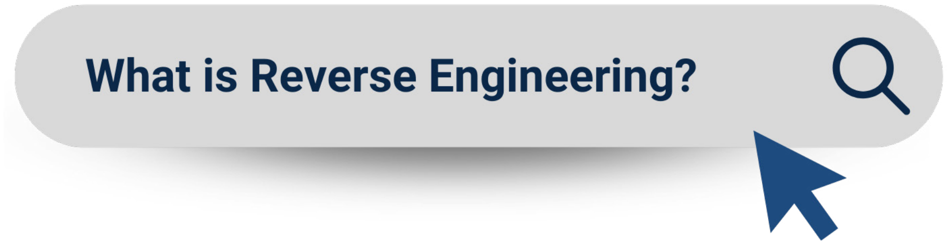
A Complete Guide



PreEmptive

INDEX

Overview



The goal of most developers is to create applications that are as functional and useful as possible. That often means collecting and using certain sensitive information from the users to provide relevant, personalized experiences.

The problem is that hackers want that sensitive data, too. They'll go to incredible lengths to figure out how to crack an application to steal data or insert new, malicious code. For instance, many hackers will reverse engineer software specifically to figure out how it works and what they can do to break its security measures.

Reverse engineering allows hackers to understand the best ways to steal information by spotting common security problems. These malicious agents look at the machine code an app generates to figure out the underlying processes. A recent study performed by Aite Group found that 97% of apps are easily reverse engineered, allowing hackers to do everything from identifying APIs to accessing and stealing sensitive data.

While reverse engineering is a significant problem, it's not insurmountable. Investing in the right cybersecurity tools can help prevent breaches of all kinds, including those caused by reverse engineering. Here's what developers need to know about reverse engineering as a threat, how and why hackers do it, and tips for protecting applications.

What Is *Reverse Engineering*?

In general, reverse engineering is the *process of looking at a finished product and taking it apart to figure out how it works*. When someone reverse engineers software, they're looking at a finished application and trying to work out what the source code is. They're trying to take it apart so they can learn how to put it together again in the way they prefer.

Most people accomplish this by using specific tools to translate the code in a completed application back into something that they can read. These tools take the machine code that makes up a finished program that runs on a computer and "decompiles" it back into human-readable code. Once this is done, the hacker can examine the code for potential security leaks or opportunities to inject their own commands.

Why Hackers Try to Reverse Engineer Code?

There are a couple of reasons why people may want to reverse engineer a program. Some people reverse engineer code for positive reasons. These include penetration testers, who are experts that explicitly want to find security flaws in an application so that they can fix them.

Similarly, "white hat" hackers may try to reverse engineer programs to stretch their hacking skills. If they succeed, they let companies know about possible flaws without actually taking advantage of sensitive data.

Many reverse engineers, however, don't have positive intentions. If a hacker targets the right application, hacking it can give them access to a lot of sensitive data that they can use to commit crimes like identity theft and ransomware attacks. For instance, if someone was able to reverse engineer a financial application, they could potentially access thousands of people's bank accounts and private details and go on to commit large-scale theft or identity fraud.

4 *Most Common* Tools Hackers Reverse Engineer Code

Understanding how hackers reverse engineer code can help developers protect their applications. Most hackers use tools like the following to reconstruct source code.

1. **Decompilers**: These tools take binary machine code and convert it all the way back into high-level source code. If a decompiler works on an application, the hacker can quickly check the source code to look for vulnerabilities.

2. **Disassemblers**: These tools take binary code and return it to “assembly” language instead of source code. This is an intermediate step. It’s harder for people to read but still allows hackers access to strings, libraries, and sensitive information.

3. **Debuggers**: A debugger can be used with both decompilers and disassemblers. It allows the user to view code one line at a time and look for potential problems or opportunities.

4. **Hex Editors**: If a hacker wants to view or edit binary code, a hex editor lets them manipulate the actual machine code running an app. They can use them to edit the application once they’ve found vulnerabilities.

Blocking these tools gives an application an extra layer of protection and users some extra peace of mind.

How Reverse Engineering Can *Harm* Businesses

But what kinds of threats can applications actually face from a hacker reverse engineering their code? Quite a few. This technique is one of the most dangerous because it allows hackers to figure out the best way to get to the data. Examples of recent reverse-engineering attacks include:

1. Attacking Specific Applications

When most people think about reverse engineering attacks, they think about “bespoke” hacks. These attacks involve reverse engineering a specific program to learn how it works, either by attacking it or generating a new program.

These attempts can be invaluable if they target the right program. A hacker only needs to successfully reverse engineer one application if that app offers enough sensitive information. Common targets of specific attacks include:

- Financial applications
- Medical websites
- Government sites and tools
- Online stores
- Gambling websites

These types of sites often contain large amounts of private or protected data. Depending on the hacker's goals, they can either steal and sell that info, use it themselves to commit other crimes, or encrypt all of it and hold it for ransom. They may even simply steal the source code to generate their own application to commit fraud by mimicking the actual site.

Reverse Engineering the NSA's Tools

One recent example of a targeted reverse engineering attack actually targeted the U.S. National Security Agency (NSA). The NSA had developed a cyberattack tool to target enemies of the state, spy on them, and insert backdoors into their applications to potentially shut them down.

However, the NSA's tool apparently did not have enough native security to protect itself. According to a recent report, it appears that a Chinese group has actually reverse-engineered the NSA's program and used the source code to generate their own trojan application under the name Bemstour. This program has since been used to target countries including Belgium and Hong Kong.

The report stated that Bemstour has significant similarities to the NSA's program. Analysis suggests that the program was created after the NSA program was reverse-engineered to understand how it worked. This kind of attack can happen to an agency specifically dedicated to security. In that case, the average business needs to be extremely cautious to avoid falling victim to them.

2. Attacking Through Reverse-Engineered Library Vulnerabilities

Not every attack goes after specific programs. Many hackers are more opportunistic. Instead of targeting a particular business or application, they look for ways they can target many at once.

Reverse engineering can support these attacks, too. The difference is where the attacker focuses their attention. Instead of a program as a whole, the hacker looks for ways to reverse engineer specific libraries.

There are a few reasons why reverse engineering a library is more effective than attacking a specific application. First, libraries usually include fewer protection measures than actual applications. That makes it easier to examine the code and look for flaws in the first place. Second, popular libraries are included in a wide variety of programs. Spotting one flaw can give them an entrance into many applications at once.

Finally, malicious agents aren't the only ones reverse engineering libraries. White hat hackers often find flaws in different libraries and post about them publicly so that developers relying on those libraries can institute extra security. However, these notices give hackers a clear explanation of how they can break into apps that haven't compensated for the flaws yet without having to do the hard work themselves.

Reverse Engineering Log4Shell

An example of this kind of reverse engineering threat occurred just this year. Ubiquiti network appliances fell under attack due to publicly known Log4Shell exploits. In November 2021, a white hat hacker posted a proof-of-concept exploit on GitHub explaining how they had reverse-engineered Log4Shell and found vulnerabilities that permitted backdoors to be installed on the devices.

Just two months later, Ubiquiti devices came under attack through onboard applications using Log4Shell. Devices that hadn't been patched to account for the exploit were taken over and had Cobalt Strike Beacon backdoors installed. These backdoors allow hackers to do everything from track keystrokes to download files and install other programs. It's unknown how much damage these attacks have done, but it's likely that victims have lost essential data and may not even know.

How to *Prevent Hackers* From Reverse Engineering an Application

Developers don't have to sit back and accept the risk of reverse engineering. Instituting the right cybersecurity features makes it significantly harder to reverse engineer code or extract data successfully. Some of the most effective techniques for preventing reverse engineering include:

Obfuscation

Obfuscation is the act of making obscuring or hiding the code in some way. More specifically, it's the process of changing an executable so that it works the way it's intended but makes no sense to hackers.

The primary purpose of obfuscation is to make it harder for hackers to use decompilers. Free decompilers are available for most popular programming languages. They can take unobfuscated code and return it to its source code in minutes. This is one of the most common ways that the average hacker reverse engineers applications.

With a solution like PreEmptive's code obfuscation, these compilers stop working. Good obfuscation can make reverse engineering a program much more time-consuming, discouraging attackers.

Renaming

Another way to make code harder for hackers to reverse engineer is through renaming. A renaming process takes the classes, methods, fields, and other named functions within the application and gives them new titles. This strips the application of information valuable to potential hackers.

For instance, a hacker looking for login credentials might look for fields labeled “user_id” or “password.” With PreEmptive’s renaming technology, these fields will get renamed with meaningless symbol strings. Hackers won’t be able to simply scan the code to find the information they need, making your application a less appealing target.

Encryption

Encryption is one of the most critical forms of security available for developers who need to protect sensitive data. Encrypting data makes it unintelligible to anyone without the key and the correct decryption program.

Developers should try to encrypt as much of your application as possible. Critical information to encrypt includes user information, especially login credentials and payment details. However, it’s possible to encrypt additional parts of the program, like the installer, to avoid having the application pirated or stolen outright.

Rooting Checks

When developing mobile applications, always build root checks into the apps. One way that hackers can access machine code to reverse engineer a program is by installing it on a rooted or jailbroken device. These devices let them use other programs that will scan and save the app’s code as it’s installed.

PreEmptive makes adding thorough rooting checks to your mobile applications easy. With PreEmptive’s hardening solution, the application will determine whether a mobile device has been jailbroken before installing anything important. If it does detect problems with the device, it will halt the installation to protect the code from malicious actors.

Integrity Checks

Hackers that are testing their ability to reverse engineer a program will often make changes to its files. These changes impact the integrity of the files, potentially inserting malicious code or backdoors.

That’s why integrity checks are a fundamental part of preventing breaches, whether they’re the result of reverse engineering or not. An integrity check is a built-in process through which an application verifies that its files have not been tampered with. Working with PreEmptive’s integrity checking system ensures that the program will only run when it’s intact and unaltered.

Protecting Source Code With PreEmptive

Reverse engineering attacks can be performed against any application. These attacks can lead to almost any other possible hack when they're successful, including massive data leaks, lost information, or ransomware attacks.

That's why it's so essential to protect applications with appropriate cybersecurity solutions like PreEmptive. Learn more about how PreEmptive can help protect your applications, or get started with a free trial today.

***Smart App Protection** for an Unsafe World!*

GET IN TOUCH:

Headquarters in USA
10801 N Mopac Expressway
Building 1, Suite 100
Austin, TX, 78759
phone: **+1 (512) 226-8080**
email: **solutions@preemptive.com**

European Sales
140 bis rue de Rennes
75006 Paris
France
Tel: **+33 01.83.64.34.74**
email: **EuroSolutions@preemptive.com**

JAPAN
AG-Tech Corp
Tel: **+81-3-3293-5300**
Email: **info@agtech.co.jp**