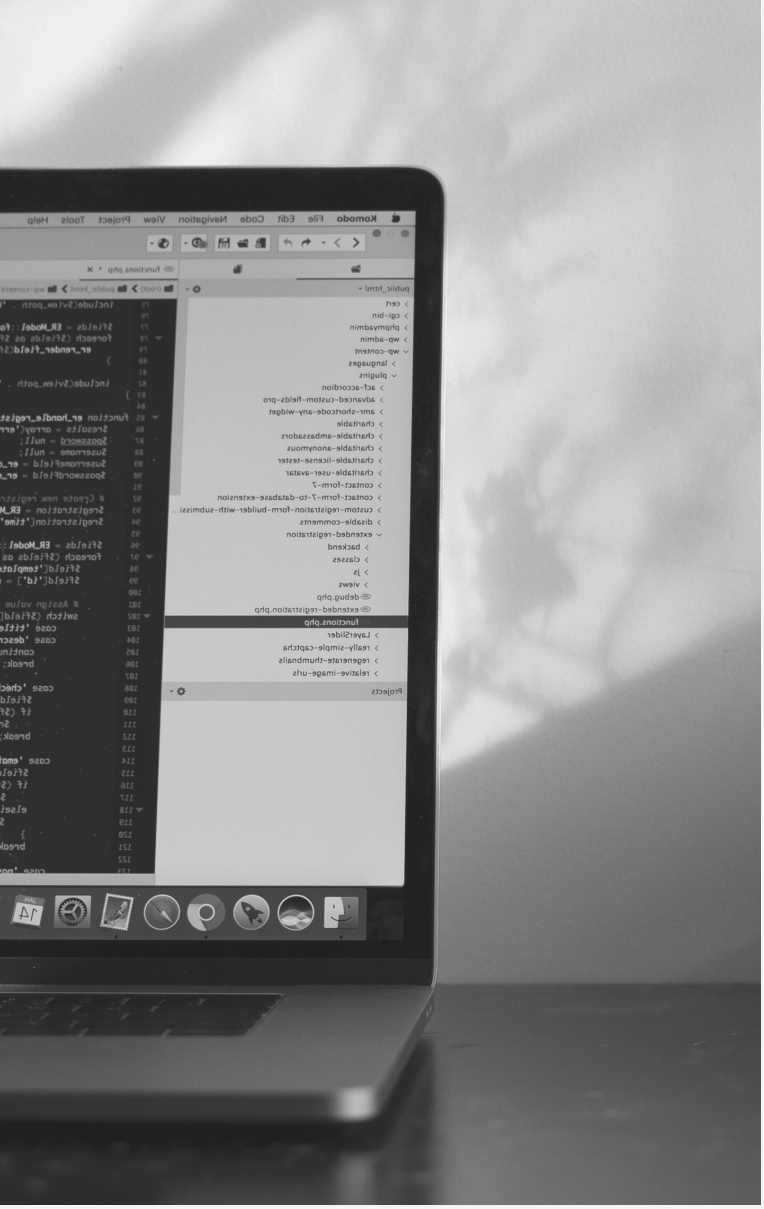
PreEmptive

FIVE FILTHY THINGS A HACKER DOES TO YOUR APP



INTRODUCTION

Anyone developing software applications today can easily feel overwhelmed by the persistent security threats their products face from application counterfeiting and malware injection to theft of services and confidential information.

This article discusses some of the ways hackers go about their dirty deeds and how to achieve a balanced perspective on application risk and risk management allowing you to release applications with greater confidence. Gaining this confidence requires a deeper knowledge of the risks and potential remedies.

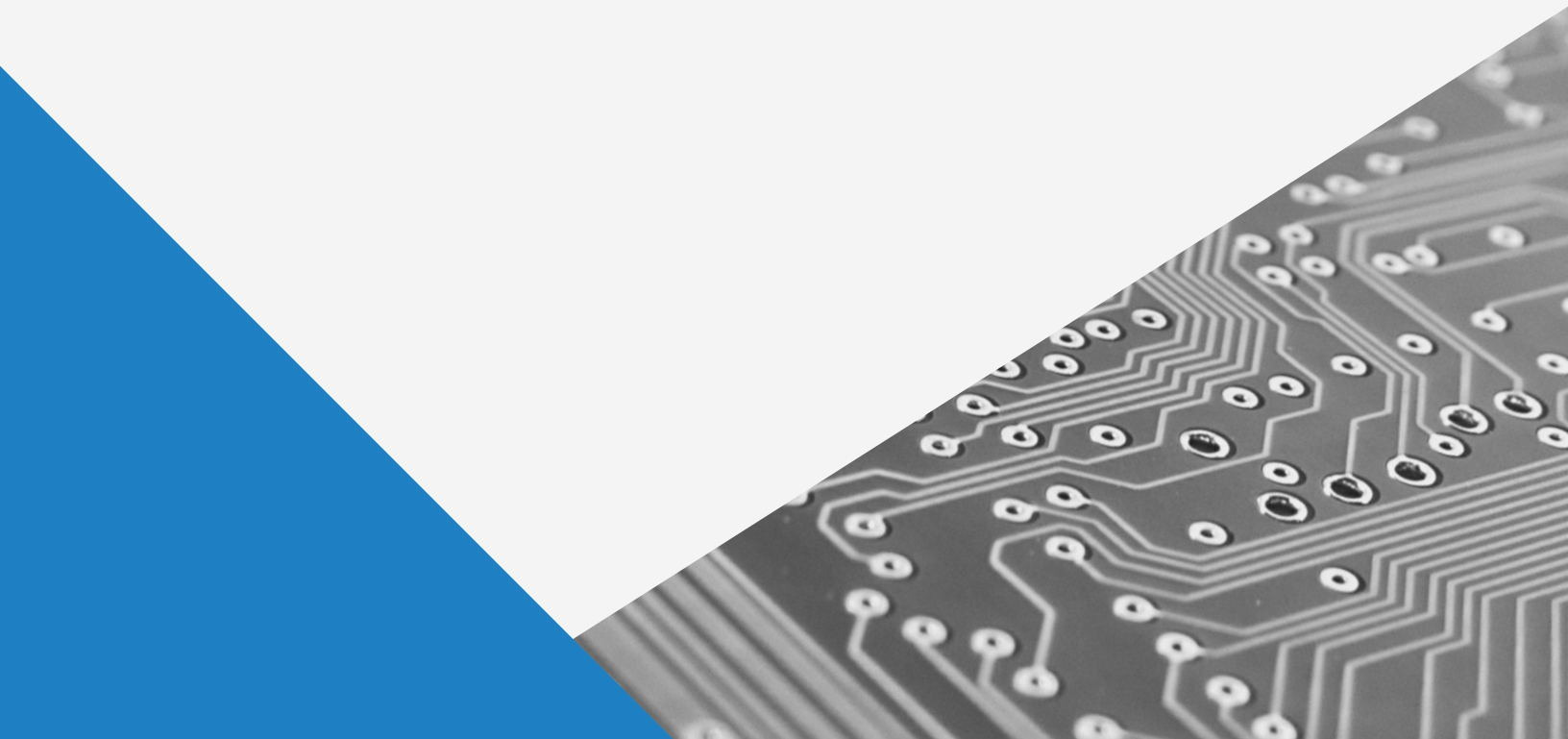
SOME TOOLS OF THE HACKER'S TRADE

A hacker might first try to reverse engineer your source code so they can better understand how the application works. Unless your application is open source there is no good reason to “give” your source code away. Let’s consider the risks associated with developing applications with managed code, specifically using .NET or Java. Chances are you’ve got some managed applications driving your business.

However, managed code is almost ridiculously easy to reverse- engineer; literally anyone can download free tools that will automatically reproduce source code from the executable in hand which can then be inspected and modified. Native applications such as iOS can also be reverse engineered, but the available native tools don’t reproduce high level source the way decompilers for .NET and Java do.

A sample of free reverse engineering tools are listed below.

PLATFORM	FREE REVERSE ENGINEERING TOOLS
.NET	dotPeek, ILSpy, dnSpy, JustDecompile
Java/Android	JD-GUI, AndroChef, APKTool, Fernflower
iOS	IDA Pro, Hopper, and r2



Having the reverse engineered source code in hand makes it easier to use debugging tools to step through code and inspect what is going on.

Some free debuggers are listed below:

PLATFORM	FREE DEBUGGING TOOLS
.NET	Visual Studio, DotNet IL Editor
Java/Android	Eclipse, Android Studio
iOS	LLDB, GDB

So, now that we know **how hackers investigate your code**, let's look at possible nefarious things a hacker might do with your application source code and a debugger in hand.

#1 - RELEASE A REPACKAGED VERSION OF YOUR APPLICATION

A hacker could make unauthorized code modifications disabling security and validation controls, bypassing licensing restrictions, inserting malware, changing purchasing requirements or ad displays in the app. They could potentially release:

- A cracked version with the licensing and validation checks removed.
- A patched version with malware inserted to steal passwords and other information.
- A copycat application on a public marketplace.

#2 - VIEW SENSITIVE INFORMATION

Source code analysis can reveal the presence of hardcoded credentials, data encryption conventions (or lack thereof) as well as sensitive information. Hackers could view the value of strings, passwords, and data utilization by your application. Depending on the nature of your application, this may be a very serious risk. HIPPA, PCI, FISMA, and other regulations recognize that data cannot be effectively managed without control over the systems that create, manage, and distribute that data. Knowledge about the internal workings of an application and the ability to modify an application's behavior can be used to access and distribute sensitive or regulated data.

#3 - SPY ON THE APP'S COMMUNICATION

If an app is running in a compromised environment such as a jailbroken or rooted device, it may be possible to spy on its communication without requiring reverse engineering or tampering of the app itself. Consider having your app detect and respond if it is executing on a compromised device. For example, the app might notify the server, show a warning to the user, trigger out-of-band additional authentication or in extreme cases not execute at all.

#4 - DISCOVER VULNERABILITIES TO LAUNCH AN ATTACK

Classic application vulnerabilities such as SQL injection or cross-site scripting do not require access to source code to discover. However, with the source code, hackers do not need to use the "trial and error" method for discovering vulnerabilities. They can examine the code directly to look for vulnerabilities - perhaps ones they never even thought to test. Once they know where the vulnerabilities are, they can exploit them to access confidential information or circumvent validation checks.

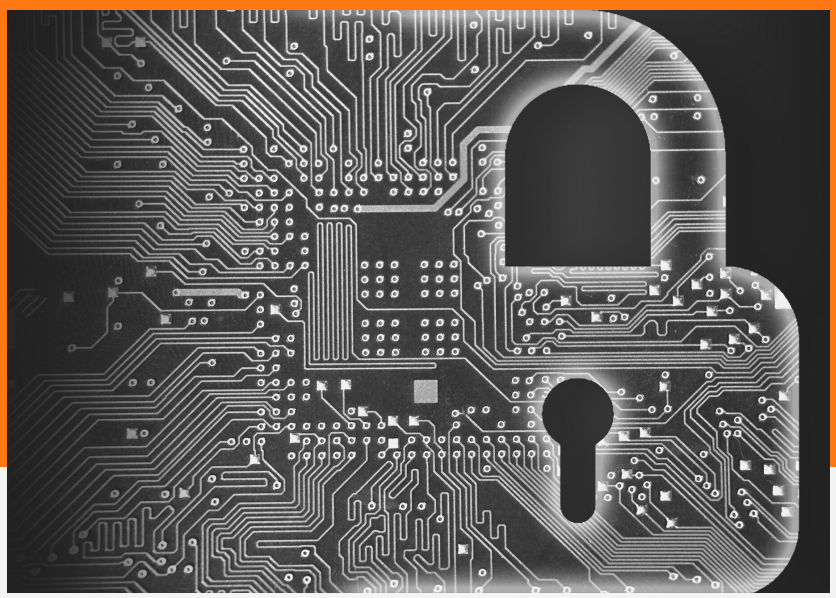
#5 - STEAL YOUR INTELLECTUAL PROPERTY

Whether you're building applications for sale, as a key part of a larger financial or manufacturing business, or as part of a line of business apps for internal use, there is likely to be IP (trade secrets) within your software. And possession of functioning source code provides transparent access to any IP that is coded within the application. So, a hacker that works for a competitor might be able to rip-off some of your technological advancement by reviewing your source code.

From a legal perspective there are three common ways to protect the IP embedded in your code:

- Patients
- Copyright Protection
- Trade Secrets

CONCLUDING THOUGHTS



Although **patents** offer the strongest protection, patenting software requires a massive certification process that is slow, expensive, and difficult. For the vast majority of software builders, a patent just isn't a workable IP protection solution. In contrast, **copyright protection** is automatic. You don't need to mark up your code, etc. and copyright law is the basis of most software licenses.

However, it comes with its own big issues; it's limited to copyrighting and distributing content. You can't copyright algorithms, innovations, or inventions, so if someone else's code looks nothing like yours or that organization can demonstrate they developed their code in isolation from yours, they're in the clear.

And with managed code, where someone else can generate the same algorithms as yours in multiple languages, copyrighting an application offers little real protection.

That leaves trade secrets, which has a lot going for them. There's no certification, they last forever, and they include concepts, innovations, etc. that give your business a financial and competitive advantage. That's why trade secret protection under the law is increasingly the regulatory strategy of choice for many development organizations.

Sounds perfect, right? Well, trade secret protection has two significant limitations. First, some major jurisdictions - like India, for example - simply don't recognize the legal concept of a trade secret. The other limitation of trade secret protection is even more fundamental: unlike copyrights and patents, it only covers things that are actually secret. Once something becomes public, it can no longer be protected under trade secret law.

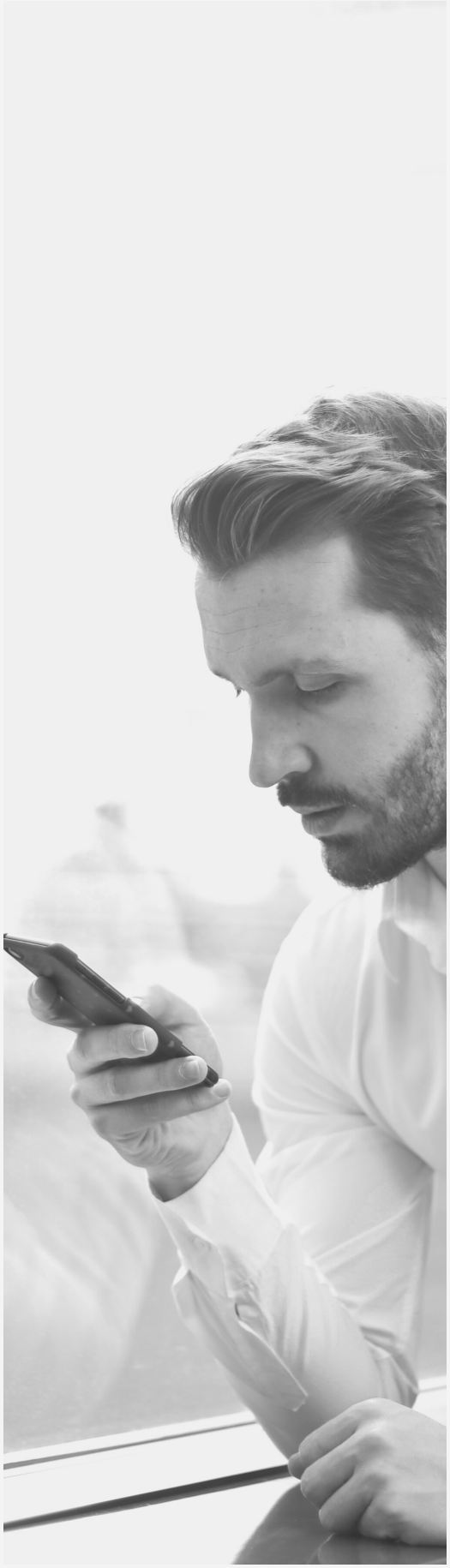
More specifically, the definition of trade secret theft requires that possession of a trade secret be achieved through improper means, such as bribery, blackmail, or espionage. Recently enacted trade secret laws, both in the United States (the DTSA) and the European Union, specifically permit reverse-engineering of any legally acquired product.

The freely available tools for managed code that make reverse engineering and tampering almost trivial have very legitimate uses within the larger development community. As with all powerful technologies, reverse engineering and application modification can be used productively, but also for mischief. A portfolio of protection technologies has emerged and matured over the past two decades and it is in use across industries. As with any risk-based strategy, there is no "one size fits all" approach, but for many applications, doing nothing is no longer an option.

To hinder the hackers, utilize a layered approach that consists of a combination of passive and active defenses. Use obfuscation and encryption to help defend apps from static analysis such as reverse engineering. And inject interlocking checks to help identify and respond to compromised devices, tampering, debugging, or emulator attacks.

PreEmptive protects intellectual property (trade secrets), data and code integrity, and ultimately reputation and revenue against piracy, theft, counterfeiting and tampering. Our layered security and obfuscation protection is directly infused into your .NET, Java, Android, and iOS applications and that means we do not require an agent or changes to your end user's computer/device or network. And our protection technology travels with your app wherever it goes. We help you manage application risks and make sure you are not making life easy for your hacking "friends."

To learn more details about how in-app protection transforms work, read "What is Obfuscation?".



Start protecting your app with a free three-day trial today.