# 50 Reasons why to choose DashO

**DashO reduces and manages risks stemming from Java application development. Our 5,000+ enterprise clients and 300,000+ users cite many reasons as to why they trust PreEmptive – and DashO specifically – to secure and increase the value of their Java applications – here are the top 50.**

## The First and Still the Most Feature Rich Java Obfuscator

DashO debuted in 1998 as a Java performance optimizer. "–O" get it? But, even as Java runtime performance improved, demand for DashO continued unabated. It turned out that the performance optimizations that we had been perfecting also had an obfuscatory side effect – in other words, our world class optimizer turned out to be a world class obfuscator – and a new category of application protection was born. Our roots in application optimization (both performance and quality centered) and our 20 year investment in application protection uniquely position DashO as the leading application protection platform.

## Security

Deterring the opportunistic, impeding the malicious, and enabling the prosecution of the successful are what application security is all about. Through a combination of preventative controls (that try to prevent bad things from happening),detective controls (that give early warning when a bad thing does eventually occur), and responsive controls (that defend against or mitigate that bad thing, as it occurs), PreEmptive secures and defends your Intellectual Property and protects against piracy, malware injection, tampering, and a host of other application and data-related risks.

1. **Strong renaming:** Overload induction takes renaming to a new level of obfuscation that not only mangles names, but destroys semantic relationships among names that can provide clues to attackers.
2. **Custom name sets:** Select unprintable characters or other unique combinations to further raise the bar against human inspection and reverse engineering.
3. **Control flow:** Our experience with optimization makes our control flow transforms effective against machine translation while minimizing performance impact.
4. **String encryption:** Our work with various defense departments and global marketplaces makes our approach secure, performant, extensible, and legal for export.
5. **Android Resource Encryption:** Hide your embedded android resources from attackers trying to inspect, copy, or alter them.
6. **Built-in "smart" obfuscation** providing optimized support for Android, Spring, Hibernate, JSP Tag Libraries, WAR files, and more.
7. **Tamper detection:** While obfuscation is designed to prevent reverse engineering and tampering, our Tamper Check detects at runtime when tampering has occurred, and can respond in fully customizable ways.
8. **Tamper defense:** Choose from pre-configured defenses (e.g. shut down the application, hang the application, throw an exception), or create your own (e.g. disable or change application-specific behavior).

9. **Tamper notification:** Send notifications in near real-time when Tampering is detected by wiring up the Tamper Check to your favorite analytics or eventing service.
10. **Debugger detection:** Attackers attach debuggers to your application in order to explore its inner workings, inspect sensitive data, and change application behavior (e.g. to bypass protections). The Debug Check can both detect if a debugger is attached to the application, and if the application is running in a "debuggable" state.
11. **Debugger defense:** You choose how the application should respond to debugging: the Debug Check supports pre-configured or fully customized responses.
12. **Debugger notification**: Send notifications in near real-time when Debugging is detected by wiring up the Debug Check to your favorite analytics or eventing service.
13. **Android Root detection:** Rooted devices may be running malware, which can jeopardize the security of your application and its data. A rooted device can also be a sign that your application is under attack, as many hacking tools require a rooted environment. The Android Root Check detects if the running application's environment has been rooted.
14. **Android Root defense:** You choose how the application should respond. Shut it down, warn the user, notify security? Whatever you can imagine.
15. **Android Root notification**: Send notifications in near real-time when rooted devices are detected by wiring up the Root Check to your favorite analytics or eventing service.
16. **Android Emulator detection:** Similar to rooted devices, running in an emulator can be a sign that your application is under attack. Unlike us poor humans, your app can really know if it's running in a simulation-- the Android Emulator Check detects if the application is running inside an emulator, giving your application a chance to enforce your policy on emulation.
17. **Android Emulator defense:** You choose how your application should respond when running on an emulator. All the standard pre-configured options are available, as well as fully customized defenses.
18. **Android emulator notification:** Send notifications in near real-time when emulators are detected by wiring up the Emulator Check to your favorite analytics or eventing service.
19. **Hooking Framework detection:** Hackers use hooking frameworks to inspect and change the application at runtime, without first modifying the binaries or running a debugger. The Hook Check can detect the presence and operation of popular frameworks, and respond accordingly.
20. **Hooking Framework defense:** The Hook Check can be configured to respond in any way, using the pre-configured responses or fully customized responses. Are you noticing a pattern?
21. **Hooking Framework notification:** Send notifications in near real-time when hooking frameworks are detected by wiring up the Hook Check to your favorite analytics or eventing service.
22. **Shelf Life**: Control your application's lifespan by making it expire at a preset or relative time. This is useful if your application is a beta, or an evaluation, or you simply don't want older versions of your applications to work indefinitely.
23. **Watermarking:** We impose structure on unstructured regions of your Jars to provide undetectable watermarking.
24. **Code removal:** Code that doesn't ship can't be attacked. DashO can remove unused code, and even remove code that is used, under certain circumstances (e.g. Android logging calls that you don't want to include in your production-ready application).
25. **Injection:** DashO's transforms operate directly on bytecode, not source, so no need to code to specific APIs. You can not only protect apps you are writing now, but any app you've ever written, without recompiling!
26. **Layered Protection:** The whole is greater than the sum of its parts. When applied together, the obfuscation transforms and runtime checks form a network of protection that is much more difficult to attack than attacking each in isolation.
27. **Timely updates:** The threat landscape is constantly changing, and hacking tools are constantly evolving. We frequently update DashO to stay ahead of the curve.

## Development Process Integration

Protection doesn't occur in a vacuum – it happens inside your development process – a process that is as likely to be purpose-built to your business as it is to change from release to release. That's why we've invested at least as much in providing reliable, flexible, and scalable deployment options to accommodate the most agile startup and the most automated manufacturing processes.

28. **Gradle integration:** DashO easily integrates into your Android or Java Gradle builds.
29. **Ant integration:** We offer a first class integration.
30. **Command line:** Of course we can be called from a command line.
31. **Stand-alone:** Updated User Interface and wizard always available.
32. **R8 integration:** Preconfigured to work with Android's R8.
33. **Incremental obfuscation**: Releasing patches? Don't want to redistribute your entire application suite? Not a problem.
34. **Stack trace translation:** Need to recover original method names from obfuscated stack traces? DashO has tools for that.
35. **Cross-Jar obfuscation:** Want to extend renaming and other advanced protection capabilities across multiple and distinct binaries? Not a problem.
36. **Distributed development support:** Need to do continuous integration or at least distributed across groups? We're built for that.
37. **Signing capability:** Need us to sign your app when we're done? It's already built in as an automated feature.
38. **Java release targeting:** Using DashO on one version of Java but you need to target another? Piece of cake – we will not break your app.
39. **Java platform support**: We offer support for Java 1.3 - 1.8, and 9-14.
40. **100% Java compliance:** From Mac to Windows, we've got you covered.

## A Risk-Based Approach

Through a combination of technology, processes and effective controls, we offer material risk mitigation lowering the likelihood of damage and transferring risk entirely out of your organization. These risks include:

41. **IP protection:** Securing your IP and your knowhow protects you, your business and your clients.
42. **Anti-piracy:** Mitigating the loss of revenue by significantly increasing the level of effort and detecting when piracy occurs saves money and perhaps your business.
43. **Social engineering:** When people know your software, they can also pretend to be you – fooling your clients into revealing private information.
44. **Malware:** Directly manipulating (tampering) with your code can be an easy on-ramp for malware unless you have taken steps to prevent it.
45. **Data loss prevention:** Attackers who know your code will be that much closer to knowing how to exploit it – don't make it so easy!
46. **Regulatory compliance:** Reverse engineering is a common, well-understood practice and the risks are well-known. If you do not take any steps to mitigate these risks, you may well be seen as an enabler (and liable), instead of as a victim.

## Licensing Model & Pricing to Support Enterprise and Mission-Critical Development

47. **Cost effective and simple to administer licensing:** PreEmptive licenses its software by the number of times a client installs PreEmptive's Software products and the number of internal users of those same products – our

pricing (and value) is directly proportional to client adoption.

What we will never do is impose an arbitrary "tax" on our clients' success, development practices, or development velocity; as an example, charging clients based upon the number of unique Android package ID's they generate violates our basic principles of fairness, transparency, and value-based pricing.

48. **PreEmptive never charges Android developers a per-app license fee** and the reasons are numerous but here are just a few:

    • **Customer focus:** Package names align with development velocity – not application risk. It's an arbitrary metric.

    • **Efficiency:** Requiring an additional step to freeze a package name, publish it to a third-party, and re-apply a third-party license prior to every build is cumbersome, time consuming, and adds needless complexity to what should be the most efficient, automated phase in an application's lifecycle.

    • **Quality:** Package name & license processing errors and/or the reuse of package names for no other reason than to avoid licensing complexity and expense make disruptions and delays inevitable.

    • **Cost and transparency:** License fees must be added to the incremental development costs with each new application and version release. Further, this "velocity tax" cannot even be reliably forecasted, making budgeting impossible.

    • **Supply chain risk:** The required disclosure of un-released product identifiers to a third party supplier can lead to unmanaged vulnerabilities and risk (counterfeiting, operational disruption …). The risk that the supplier may not be able to generate an updated license within a critical response window adds an additional (and wholly unnecessary) service level dependency.

    Any supplier familiar with enterprise software development priorities and practices would understand how disruptive and counter-productive this kind of pricing policy would be.

## Customer support

Post-compile protection of your application is no small task and typically happens at the very end of your production lifecycle – where time (and a sense of humor) are often in equally short supply. That's why we focus on managing your risk and accelerating your work with a hands-on approach.

49. **Dedicated customer support staff:** Our dedicated, professional team knows how to program, they know our technology, and their entire job is to make you successful.
50. **Live support:** Our dedicated team will actually chat or (heaven forbid!) speak with you live on the phone.
51. **Enterprise-ready SLA capacity:** We have the ability to commit to organization-specific SLA's allowing you to actually transfer risk from your organization.
52. **Online support:** And we post our (and our clients') tribal knowledge online so you don't have to reinvent the wheel.

## The PreEmptive Difference

Over the past 20 years, our team has focused on hardening and protecting high valueapplications. With over 5,000 clients and software on millions of developer desktops, there can be no question that organizations that are serious about building and securing the best software possible trust PreEmptive. Here are a few final reasons why.

53. **Category creators:** We don't just lead in the obfuscation and app risk management markets – we literally created them.
54. **Proven quality and service:** The largest manufacturers, life science companies, aerospace corporations, financial institutions, and (of course) software development organizations all trust PreEmptive to secure and measure their work without compromising the quality and functionality of their code.
55. **Cross platform:** From Android to Amazon – from iOS to .NET – to the next great thing – we support the frameworks, platforms, and surfaces on the day they arrive – we will never hold you back.
56. **Exceed expectations:** Just as we delivered 56 reasons to work with DashO when we only committed to 50 – we will exceed your expectations for quality, responsiveness, and capability.