While there are many vulnerabilities that come with traditional antivirus (AV) products, such as simple "detect and respond" strategies that allow threats to execute and large agents that suck up vast amounts of endpoint resources, there is one vulnerability that often goes unnoticed – the weaknesses of the antivirus engines that drive these traditional products. Cylance's engine does not have these weaknesses.

Many traditional AV products come with engines that have a significant amount of code that dates back 10 to 20 years, long before the types of problems that lead to things like memory corruption vulnerabilities were well understood. Additionally, old products have a "swamp" of old code that has accreted and has little value for modern use, but may still pose a serious threat. Generally this older code isn't well tested or maintained, as it applies to features that people generally don't care about. However, as it still exists, it becomes a code path for attackers to exploit.

A Different Approach

But Cylance took a very different approach when building the engine that drives its solutions – the code was built from the ground up. First and foremost, Cylance treats code security as an important design decision, not something that can be sacrificed to save time by building on an existing, vulnerable framework. Many of the original team behind Cylance came from a background of vulnerability research and have led product security teams in traditional AV spaces, making them well aware of the threats inherent in recycling old code to build an AV engine.

Cylance products have no swamp of old code and were built with a deep understanding of the types of attacks that face modern software. Cylance fundamentally understands that a defensive product like an antivirus engine is, in reality, also an attack surface. The whole purpose of a defensive engine is to take potentially attacker-supplied inputs and make a decision on good and bad. In addition, for the product to be useful, it's often placed in line of nearly every type of input. This can include mail gateways, network paths, and of course, endpoints. Since the value comes from not having to explicitly make a determination, it's designed to be transparently inline, meaning that attackers have a huge number of vectors to reach the potentially broken code. With that in mind, hardening the entirety of the product is a necessary primary concern.

Classic Vulnerabilities Overcome

There are three basic classes of vulnerabilities that can apply to an antivirus engine. First and perhaps most dangerous are memory corruption vulnerabilities. These are the traditional stack, heap, and integer overflow type bugs. These very often can result in code execution, regardless of "advanced protections" in place, such as DEP, ASLR and compiler flags. Next up are logic bugs. These include crafted inputs that can result in an unintended consequence without memory corruption, such as command injections, branch avoidance and crafted infinite loops. Finally, is the

old standby, denial of service attacks. By sending an input that can cause the engine to work extra hard, attackers can bog it down, run it or the underlying machine out of resources, or even disable the engine entirely.

Cylance's engine was designed with state of the art technology and the history of attackers (knowing well that attackers frequently dredge up attacks of the past) in mind. The Cylance engine design focused on three key areas; hardening against direct attack (sending crafted inputs directly to the engine), hardening against indirect attack (avoiding the engine using various techniques), and focusing on performance elements to avoid DoS attacks.

Direct attacks are thwarted by a pure .NET approach to building all engine components. By using a purely managed language, many of the traditional bugs are taken off the table. A variety of managed languages were considered, but .NET was chosen because of it's relatively clean security history, it's ability to be portable through standards and mono, and it's performance profile, which is close to native c++ when used with a modern runtime with JIT enabled. By using .NET in all operations dealing with user-supplied data, the Cylance engine avoids many of the pitfalls that affect other engines. Additionally, the .NET approach enables a significant reduction in the number of odd code paths that many others have to support, including ancient and unused features of windows or formats, which can simply be blocked or scanned after they are in play. On the logic bug front, the engine focuses on correct exception handling and knowing when it is being taken for a logic excursion. This helps the engine break apart many samples intending to cause harm. Additionally, since the engine doesn't consider any input to be trusted, it uses significant amounts of validation on inputs and avoids making unbounded assumptions.

Indirect attacks typically revolve around the concept of "Self Protection". The Cylance engine stops as many paths to disable or remove services as possible (following guidelines such as the Matousec self-protection tests), and focuses efforts on looking for any type of bypass attacks. A combination of memory defense and the engine itself help prevent many types of attacks that could do the engine harm.

Finally, on the DoS side, the engine has internal metrics and components in place to reduce the chance of a runaway or faulting system taking the engine offline. Cylance conducts both spot and holistic performance testing on every release and processes hundreds of millions of samples through an engine prior to release, allowing us to identify nearly any pattern of performance degradation. Outliers in performance raise alerts within the backend-processing infrastructure and are used to further refine performance metrics. This testing hardens the Cylance engine against both known and previously unknown threats before they can execute.

About Cylance:

Cylance is the first company to apply artificial intelligence, algorithmic science and machine learning to cybersecurity and improve the way companies, governments and end-users proactively solve the world's most difficult security problems. Using a breakthrough predictive analysis process, Cylance quickly and accurately identifies what is safe and what is a threat, not just what is in a blacklist or whitelist. By coupling sophisticated machine learning and artificial intelligence with a unique understanding of a hacker's mentality, Cylance provides the technology and services to be truly predictive and preventive against advanced threats. For more information, visit cylance.com

+1 (877) 973-3336
sales@cylance.com
www.cylance.com
18201 Von Karman, Ste. 700, Irvine, CA 92612

CYLANCE