# All.Net Analyst Report and Newsletter

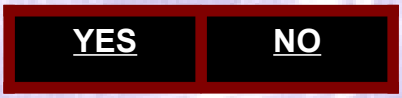
***Welcome to our Analyst Report and Newsletter***

**Eventually, you are going to make a mistake**

Nobody is perfect. No matter how good you are at what you do, no matter how careful, at some point, you will push the wrong button. That's the theory of the attackers sending out an unlimited stream of attack mechanisms that exploit different weaknesses in systems offering you choices you should never have to make. The idea of choice is a mistake in most cases, and the exploitation of choice is one of the most prevalent methodologies underlying attacks today.

**But isn't choice good?**

Free choice – it is the fundamental of the free society. And yet choice in computer security is almost never a good idea. The choice users are given, time and again, is whether or not to allow access by a program. Eventually, any user will enter the wrong answer. And of course, the answer is often unclear because the question is often unclearly stated. Do you not agree?




By the way, some decision will pop up with slightly different wording and different meanings to the possible answers many times per day on average. Even if your answers are right 99% of the time, that means that every few weeks you will get one wrong.

**Get it wrong once...**

and the attackers will gain access to the account you are using. The fundamental problems here are many-fold and have gone unaddressed for decades.

- The issue of user choice is obvious by now. Enough choices and eventually you get one wrong, and the consequences may be ongoing arbitrary unlimited remote access.
- The second major problem is that once you get one wrong, whatever they did will remain indefinitely, because there is no obvious way to tell what the side effects were after the fact, and even if there were, getting rid of them is essentially infeasible.
- The next major issue is that the available countermeasures are not well supported. For example, isolation of such things is incredibly hard and inconvenient in today's computing environments.
- Another major problem is the lack of trusted path. Trusted path has long been a known problem, and it is fundamental to trusted systems. The basis problem we face is that we cannot tell the difference between a legitimate request and a Trojan horse request. This should be immediately obvious on its face to every user in every system.
- And then there is the problem that most users use one account for a myriad of things. Once it is taken over, essentially everything they access is accessible by the attacker. Switching accounts is not very easy to do, and it creates lots of inefficiencies.

There are other similar problems, but they all come down to the same thing. Efficiency, ease-of-use, and utility are at odds with the poor security implementations we are using today.

**And it will happen more and more**

Its getting worse every day. And it will keep getting worse unless and until the sellers of the computer systems and software we use are forced to do better. The market won't do it. That's just not how markets work. It costs more, makes it a bit harder, and the market for “secure”, even if it exists, won't have the opportunity to get expressed.

Ten years ago, a pretty decent expert could get Linux to operate securely with substantial effort. While Windows and OSX and other operating environments in widespread use were insecure and not readily securable, the automatic update process has made the entire ecosystem problematic and full of uncontrolled software changes from afar. Linux cannot be reasonably secured without a great deal of effort, and Windows and OSX are pretty much hopeless. The extension of trust to third parties has become so untenable that it is infeasible to even know who wrote the code you depend upon or how many parties were involved. The notion that review will find holes in code was long ago a fantasy. When scores of previously unrevealed vulnerabilities appear in major utilities upon a few weeks of testing, the situation becomes nearly unmanageable for the average, or even sophisticated end user.

**Where do the users go from here?**

The most tenable solution I have identified for the motivated end user wishing to be efficient and prosperous in the greater Internet environment today is a plan of risk disaggregation using virtual machines and remotely mounted (via ssh) encrypted disk areas within a platform that is relatively locked down. A notional version of this is operating in some of our internal environments today, and while it is a bit inconvenient, it is reasonably functional, until people start making big rather than small mistakes. They will still be made, but with less frequency and less consequence. Here's the basis outline of a plan:

**Virtualization:** In each user machine there are a set of virtual machines (VMs) connected by the raw computing environment, which is limited in its use. For example, suppose we have OS-X as the raw environment, and within it we only do the basic network connectivity, file system management, remote access management, display management, and similar activities. Within this environment, we run a set of virtual machines for different partitions of different functions with different operating environments. For example, email is constantly used and too inconvenient to separate out. Even though we have many email addresses per person, they are aggregated into merged views by the email client, which resides in its own VM. But many less universally useful operations, like software development and testing platforms, are kept in their own VMs running Linux, Windows, etc., that are booted as needed and shut down when not in use.

**Partitions:** The partitions, in this case operational partitions, each run with remote access to related encrypted disk storage areas, such as project areas and similar data sets associated with the individual and their uses. Each partition may be accessed from multiple virtual machines, and all have more or less equivalent access based on user ID and partition. For example, we might have one partition for each major client, project, or field of endeavor, and some other more generic partitions for things like Web sites, etc.

**Operation:** To operate in this environment, the user typically starts some standard VMs (from read-only images) at login, such as the email VM, and mount their storage partitions. These are rarely shut down. When performing a specific task, such as accessing sensitive

databases for a project, a VM is started (from read-only storage), disks are mounted from within the VM, and appropriate programs are started up. When work on a project starts, a 15-30 second overhead occurs, but from then till the work stops, operations proceed as usual. Typically, these processes operate for minutes to hours, depending on specifics, but when they are no longer in use, for example when the current draft project report is done and sent off to the customer, VMs are shut down, partitions are unmounted, and content is no longer accessible. Moving content between partitions requires a VM authorized to access all relevant partitions, or a system of drop boxes can be arranged as temporary storage for movement from and to partitions. Thus you drop files from a project into a partition to email it, and email clients can drop files into storage readable by VMs. The time costs are nontrivial, but it is a tradeoff that can balance time with risk by having smaller or larger partitions and VM contexts.

**Backup and restore:** To back this system of mechanism up, an internal backup server with no direct external access, mounts partition after partition, doing backups to internal disks drives. These are rotated into secure storage areas periodically. Restoration uses the same mechanisms – remote mount and copy of the backup into the partition. This is all automated so that a single command does a backup of all of the VMs and partitions, and this can be run periodically if so desired.

**Limitations:** Of course all such systems have their weaknesses. For example, the master system where the user resides could be attacked and used over time to gain increasing access during the periods of operation of the VMs. Remote control could be used to activate the VMs when the user was not present. The main security mechanism in this approach stems from protection of the environment used by the user, and it's easy to make a mistake of the desired applications are present in that system without using a VM. Updates to the controlling system are still problematic, and of course each of the controlled systems potentially gets more and more compromised with time. If the VMs are rebooted from read-only media each time, they are not readily updateable, but on the other hand, any attacks will not persist past the reboots that they more frequently experience. Common mode failures, like successful attacks on the cryptographic infrastructure, and of course attacks on the controlling system, can potentially defeat all of the other controls. This is only a start, but you get the idea. And the biggest limitation of them all – it's too complicated for most users today.

**Tradeoffs**

Users working in today's “modern” computing environments are practically guaranteed to be successfully attacked. The alternative is to live in the past. But there are ways to better control risk and live in a mix of the past and future – until and if ever things change. Tradeoffs in terms of reduced ease of use, speed of starting key operations, ability to rapidly integrate any content from anywhere with any other content from anywhere else, and similar things must be made in order to disaggregate risks, reduce the consequences of most attacks, slow attackers, and gain and retain control over the situations most likely to do the most harm.

But most users don't know how to control these risks, what the tradeoffs are, and how to apply them. And worse yet, implementing such controls is not readily facilitated in a secure manner by available mechanisms or made automatic in normal user experience. If these sets of controls were automated and transparent to the user, the story would be a lot different, but for now, only trained and careful users can even attain this level of control while still operating in modern computing environments. And eventuality, we all make mistakes...