TrustWrx, Inc.
Threat-immune Security for
the Internet of Things

11 Internet Security Fallacies

The Security Space is Beset with Fallacies that Impede Much Needed Privacy Innovations

There are many fallacies and myths that currently dominate the security/privacy industry; severely limiting the much-needed innovations towards better privacy and security. These beliefs have evolved over time and are well entrenched across almost every nook and cranny of the present security marketplace. Most of the fallacies are taught across all levels of academia, are sustained by well-meaning security engineers and marketplace pundits, and absorbed as gospel by a naïve public.

These fallacies are reviewed here so we clearly understand the realities of the market and can plan our product and marketing strategies accordingly. Many privacy/security solutions, and many security consulting companies, have been built on the ready acceptance of these fallacies which has left them weak and vulnerable.

These fallacies dominate the conversation in the security space, while the more important issues, like really good key management and locking down ports, remain sotto-voce. It is important to note that of the top major hacks of 2018, none were the result of discovered keys or a direct cryptography attack. Poor security hygiene dominates, along with unpatched software, unhygienic coding practices and compromised applications, and a few other large-scale database spoofs.

Achieving privacy is a critical requirement to achieving real security. Today one can buy a lot of security and not get much privacy. However, if one achieves real, durable communications privacy, much higher levels of security will be the result.

11 Fallacies in the Security/Privacy Industry

1. **"Content encryption equals communication privacy"**
2. **"The security industry has dominion over the privacy problem."**
3. **"Encryption itself is a solution to security and privacy concerns."**
4. **"Large bit-depth cryptography is better than small"**
5. **"Everyone knows that obfuscation doesn't work."**
6. **"The security agencies can crack encryption."**
7. **"Quantum Computing will crack encryption"**
8. **"Multiple layers of encryption don't improve security."**
9. **"Cryptography Solutions must be open source to be trusted."**
10. **"The public DNS is the only way to route and is sacrosanct."**
11. **"Security should be baked into the application layer"**

The following fallacies are not presented in any order of importance.

- **Fallacy No. 1:** **"Content encryption equals communication privacy"**

Encrypting only the message body and attachments does not satisfy the need to protect a message from prying eyes. In today's world the packet header is the much more sought-after element - it is the only thing most hackers care about, not content. The header is much more important to government surveillance, which is far more interested in who is talking to whom, while what they are saying is purported to have value, but that has yet to be proven. This is why current laws let the intelligence community store headers indefinitely, and why they now want to compromise encryption.

This fallacy survives because of the legacy belief that the message header must be unencrypted and openly visible for the message to route over the public Internet. This is valid for the insecure public side of the Internet where visible and DNS-based routable headers are the only routing available. For secure communications, TrustWrx has solved this problem by routing privately in the application layer and triply encrypting every part of the transaction, including routing headers.

Routing messages publicly and expecting message privacy is a mutually exclusive conundrum.

- **Fallacy No. 2:** **"The security industry has dominion over the privacy problem."**

On the surface it seems logical that the security engineers on the front line would be the logical providers of privacy solutions. Exactly the opposite has proven to be true. The security space has carried this ball to date, and the losses continue to mount. Why? Security engineers are hampered by working in a dark culture dominated by a fortress mentality. Their training and their world is one of combat; of gates, guards and guns (while many engineers see it only as a big video game.) They spend their careers hoping to win the occasional battle in a war they can never win.

Vigilant in their network trenches, they erect barriers, scan for threats, and end up playing whack-a-mole against zero-day attacks that just keep on coming. Consequently, the security industry has evolved as a rather myopic space focused on militant-like defense.

Moreover, we often see many security engineers who perceive that their livelihood depends on having the problem, not solving it. We have learned that a meeting with security engineers is doomed to status quo arguments, so our sales focus is at the business VP level and above where the bigger privacy needs carry more weight. The myriad defensive problems security engineers deal with will never go away - their jobs are not actually threatened by messaging privacy solutions - but the defensive and protective mind set of that cohort will remain a challenge to TrustWrx.

Security engineers are warrior gatekeepers and fortress builders - not guardians of the fair maiden of privacy - and certainly not qualified application software developers.

- Delivering privacy solutions requires a much more sophisticated and proactive application approach in the application layer of the OSI model. Providing the myriad set of elements that IoT privacy demands involves a well-integrated application product, embodying encrypted databases, multiple-layer message encryption, port-knocking, key management, certificates, etc. – all bundled into a comprehensive application layer solution. This is a world significantly broader from the narrow battlefield and fortress mentality of the current security industry.
- **The much-needed privacy industry does not yet exist.** The problem is that many think that it does; confusing privacy with security. There are many privacy policy web sites that dance around the concerns, and privacy conferences that are mostly policy oriented, but little to no technology focused market awareness that privacy is quite different than security. t.

- **Fallacy No. 3:** **"Encryption itself is a solution to privacy concerns."**

Encryption is at best an enabling technology, never a solution by itself. Standalone encryption is the domain of those with sophisticated cryptography skills. Managing key generation and exchange, SSL certificates and the encryption/decryption process, along with the many related complications, is beyond the skills of the average user. For every two competent users that can manage their own message encryption, there are a million users/devices that need a non-technical packaged solution.

- **Fallacy No. 4.** **"Large bit-depth cryptography is better than small"**

Most in the cryptography community believe that strong levels of encryption are important when there is actually no evidence anywhere that large-bit keys – above 128 - make any difference at all – other than a possibly legitimate legal defense.

As proof that this presumption is fallacious, we only need review the history of secure web sessions. Almost all online secure web sessions (HTTPS – RFC 2660) operate on standard AES-128 encryption – a key of 128 bits - first introduced in 1999. Since then it has successfully protected all sorts of financial, medical, commercial, government and personal information worldwide. The cryptography community occasionally reports a minor vulnerability (i.e., the Open SSH coding flaw), but these vulnerabilities have yet to result in any reported individual or large scale penetrations of HTTPS transactions.

To date there have been NO reported direct hacks of HTTPS web sessions.

One may conclude that any encryption at all (above 64 bits) is sufficient to deter casual, criminal or government penetration attempts of any vigor. Regardless, cryptography geeks continue to promote key size recommendations up to 2056 bits, simply because bigger guns are perceived to be better than smaller guns. *This is similar to the red sports car myth. Red cars don't go faster than any other color, but the owner does let folks know he is driving a fast car.* Keep in mind, also, that larger keys require significant more CPU cycles, which is not feasible in the resource-sensitive IoT space.

Only academics and spies waste their time on brute-force cryptography attacks, and those discussions are almost all theoretical. (A brute-force attack uses very high capacity computing resources to guess at keys or factor a prime number.) Modern encrypted content is decrypted primarily because the keys are disclosed or discovered, which renders meaningless the much-touted large bit-depth cryptography/key argument. Any advanced cryptography system must presume that the keys will be discovered, and the system should provide the mechanisms that render that discovery ineffective to the attacker. (The TrustWrx triple-layer encryption module and automatic key management was designed to solve that problem.)

- **Fallacy No. 5.** **"Everyone knows that obfuscation doesn't work."**

This myth survives from early and mostly failed attempts at obfuscation that were typical during the birthing days of the Internet. We have heard this generalization repeatedly from security engineers. We point out that encryption is itself a very strong form of obfuscation. The obfuscation questioned here is the manner in which TrustWrx fragments message exchange house-keeping, key exchange and encrypted content across multiple packets that are multi-layer encrypted.

The TrustWrx solution takes advantage of the extreme difficulties of finding among the packet storm the right packets, cracking three layers of encryption per packet, and relating inter-dependent keys and pieces of a message, across packets sent separately to varying IP addresses. Those difficulties are characterized by detractors as obfuscation, which indeed they are – but extremely strong obfuscation that no independent testing lab was able to penetrate. Furthermore, it is well accepted and considered irrefutable that there is no way to exchange keys and content privately over the Internet without sophisticated obfuscation of packet traffic.

The well-crafted distribution of keys and content across multiple packets, in combination with durable layers of encryption, results in obfuscation that is sufficiently complex and sophisticated that it does thwart even the most determined attack, including a quantum-based attack. More notably, it presumes that a determined attacker might discover bits and pieces, but the complexity is such that partial discovery remains of no value.

As a side note: revealing this construct often provokes a discussion about performance. A comprehensive analysis of the TrustWrx message engine in 2009 at NSS Labs in Austin, Texas resulted in a high-performance carrier-class rating. The test was conducted in a low-latency fiber lab on two low-end Dell 1435 servers, on a network configured for 1,000 desktop clients. It produced more than 6 Terabits per 8-hour day of throughput on 1 MB-sized messages. In normal day-to-day operations, users experienced no discernable delays in sending and receiving private messages. *The NSS report is available upon request.*

- **Fallacy No. 6.** **"The security agencies can crack encryption."**

The post-Paris attack anti-encryption hysteria – driven by the intelligence community's efforts to enact backdoor legislation - has pretty well laid this argument to rest. Why do they need legislation to get backdoors if they can crack encryption? Regardless, the academic and cryptography communities continue to ride this lame horse with speculations aplenty. Online, one can find endless sites and blogs that over-work the theoretical mathematics and analytics by which cryptography might be cracked. There is endless speculation that aggregating a few hundred powerful computers or employing Quantum computing (which does not commercially exist) would crack AES-128, and it would only take a few hundred years to crack the first packet. Notably, there is not one instance presented anywhere of a successful method by which encryption, of any bit depth above 64 bits, can be decrypted in a timely enough manner to be of any value.

There is only one conclusion. For all practical purposes, nobody can crack modern encryption.

Asking for backdoors means asking for the keys by which plaintext is encrypted. Rather than brute-force mathematical attacks on an encrypted file; it is far more efficacious to go hunting for hidden keys, and they are much easier to find. Of a standard PKI key pair, the public key is easy; it is usually advertised openly. The private key is usually hidden somewhere on the computer or the server and can be discovered in transit or at rest with some work. Once the PKI keys are known the packet can be decrypted – presuming it is single layer encryption.

- The NSA, CIA, FBI and GCHQ have all tacitly admitted that they cannot crack encryption. Previously, they have achieved partial success by weakening commercial solutions with dark subterfuge (the NSA/RSA fraud, Juniper Networks, etc.) Most recent editorializing on the web now lays on nation states the blame for the recent DDoS attacks against encrypted message services. In other words, the government agencies – unable to crack encryption and unsuccessful at legislation - now appear to be attempting to cripple the use of private message services worldwide by overwhelming them with DDoS attacks.

- **Fallacy No. 7 -** **"Quantum Computing will crack encryption"**

This fallacy is a child of the major promise that quantum computing is many orders of magnitude faster than standard CMOS computing. The immediate assumption has been that with all that speed encryption keys can be quickly guessed or a brute-force attack can succeed. Quantum computing remains largely theoretical and is unlikely to be a real threat to encryption because quantum mechanics is inherently probabilistic rather than deterministic. Encryption is a 100% deterministic process that requires that every bit remain intact, which is not possible in a probabilistic state.

The one "Quantum computer" on the market today is an early stage and rudimentary version (actually a quantum annealer) without the ability to crack encryption. It is priced at US $15 million and operates at vacuums and temperatures more stringent than interstellar space. Furthermore, it must rely on standard CMOS memory and circuitry at normal speeds which throttles performance drastically.

If a true quantum computer is ever built with an adequately sized qubit processor it will still not threaten the three-layer architecture of the TrustWrx solution. Layered encryption will defeat any direct brute-force attack either through key-guessing, key discovery or integer factoring.

Regardless, new quantum immune encryption algorithms are already in the works and will be standardized and introduced long before quantum computing hardware becomes commercially viable.

- **Fallacy No. 8.** **"Multiple layers of encryption don't improve security."**

This fallacy is perpetuated by theoretical discussions online cautioning that multiple layer encryption, if done wrong, will make the attack surface highly vulnerable, which is correct.

The proper approach is to cipher the original text with one algorithm and key set, then re-cipher with a different algorithm and a fresh key pair, which renders the attack surface far more complex. There is a third more advanced approach, which TrustWrx employs in an extended form. This approach is illustrated in an early article on layered encryption. This method exposes different parts of the encrypted message at different decryption levels, protecting the content from being revealed, while exposing the encrypted routing and housekeeping information only where needed.

The other main virtue of multiple layer encryption is the extent to which it thwarts even visibly revealing when the outer layers have been successfully cracked. Successful decryption returns only readable plain text. If one does discover or guess the right primary level keys, the TrustWrx decryption process will return a lower-layer presentation that still appears to be encrypted, leaving the attacker to believe that he purloined or guessed the wrong keys.

The purpose of multiple-layer encryption is to protect against the encryption keys being compromised; it is not to make a brute-force cipher attack more difficult. By far the weakest link in the entire cryptography process is keeping private keys secure. A durable cryptography system must be designed with the expectation that the PKI key pair **will be found out**. If the key pair is discovered and then used to decrypt layer two, and layer three is encrypted with a different cryptography, the attacker has gained nothing and must start over. The trick is managing the key exchange for layer three – a one-time pad symmetric key - such that there are no cross-references in the packet storm behind which the various keys are protected. This is neatly accomplished by transmitting keys in separate packets using the same one-time pad and multiple layer technology. This also leverages the packet storm dynamic to obfuscate any possible discovery of the complete chain of keys and content relationships that make up a secure message in transit.

Multiple layer encryption can only be accomplished by constructing an application layer solution that bundles the necessary components, utilizes secure databases and multiple open source encryption algorithms, properly manages key generation and exchange, handshakes and verifies the sending device fingerprint, and does it all transparent to the end-user. To complete the belts and suspenders approach, all TrustWrx packets are sent under authenticated protocols within a standard IPsec or TLS session (AES-128 or 256), which provides the outer layer three cryptography protection. *Many of the complex elements discussed above are protected by the Sidman patents, first filed in 2004.*

- **Fallacy No. 9.** **"Cryptography Solutions must be open source to be trusted."**

This myth comes from the early days of encryption, when cryptography algorithms were emerging with as-yet undiscovered vulnerabilities. Broad scrutiny by the white hat community did succeed in reducing vulnerabilities and remains today a valid and valuable oversight to the advanced algorithms in use. And, everyone agrees that a closed source cryptography algorithm remains suspect and is to be avoided.

Over time that generalization of open scrutiny has been erroneously expanded to include the idea that all cryptography or privacy applications must be published as open source if they are to be trusted, which is faulty. It is true that the encryption algorithms must be open source, but revealing the application client/server source code; the methods and details of secure database structure and management, file names and storage locations, key generation and exchange methods, and the many other critical components of a truly secure application solution would reveal everything to all eyes and would render the overall solution completely untrustworthy – and thus be fatal.

> **Understanding that, we may conclude that all encryption algorithms need to be well-scrutinized industry-standard open source, while the application layers that manage privacy must be proprietary.**

This does not mean that we shouldn't reveal the multilayer encryption of TrustWrx in general terms - both as a strong sales message, and a discouragement to any hacker that might see cracking the TrustWrx technology as a worthwhile challenge.

> As a key historical observation, keep in mind that that RSA placed its patents in the public domain in 2000. The "source code" of RSA's Dual Elliptic Curve cryptography was also published and was widely adopted across the globe by many corporations and government agencies. Until Snowden, nobody knew that in 2000 the NSA had paid RSA $10 million to backdoor RSA's technology, with the purpose of placing a corrupt technology into the marketplace that did exactly the opposite of its advertised purpose. The published source code hid the backdoor (BSafe, a closed-source cryptography algorithm, approved earlier by NIST) while the distributed runtime code utilized it. As a result of Snowden, Dual Elliptic Curve was generally discredited, the RSA Conference in 2013 (the major worldwide security conference sponsored by RSA) was nearly wrecked, and thousands of enterprises and governments worldwide scrambled to remove the formerly popular RSA cryptography from their systems.
>
> The published source code fraud on the public by RSA and NSA rendered the publishing of source code, to a certain degree, suspect and discredited, in spite of the value it brings to cryptography and other technologies. It revealed that published code is not necessarily run time code.

Fallacy No. 10. "The public DNS is the only way to route and is sacrosanct."

The public Domain Name System (DNS) will always have a dominant place in routing public services (online web sessions, social media, email, public messages, etc.) and will always be susceptible to unwanted traffic and malware. The public DNS is somewhat more than a simple phone book for domain names and IP addresses. On the surface it appears to be a critical resource so we don't have to remember numerical IP addresses. But, the public DNS is an open, unencrypted,

unauthenticated public resource with numerous vulnerabilities that make it one of the worst enemies to privacy. However, there is no need for a medical device or grid transformer to publicly advertise its domain name or IP address; it wants a fixed relationship with known entities, and no one else should be allowed to see its relationships revealed through public routing.

The heart of the public DNS is the Resolver that links names to numbers. The Resolver allows open visibility to all records, direct and reverse lookups and it provides other rich info that hackers exploit. Over the past ten years attempts have been made to "secure" the DNS with DNSSEC (RFC 4035 – March, 2005). These revisions provide encrypted authentication for updates to the DNS records, but provide no confidentiality for its use. Adoption of DNSSEC has been sparse as backward compatibility problems have not been solved and unless the large majority of the Internet utilizes DNSSEC any normal lookup would more likely resolve to standard DNS.

It is universally believed that the public DNS, in spite of all its problems, is sacrosanct. It is true that the public side of the Internet could not function without the DNS. Defaulting to DNS references is what makes message, web surfing, social media and other online addressing work in the public space. It is attractive because nobody needs remember an IP address; it makes programming easy, and it allows the IP addresses to change without having to change client-side domain naming. However, the IP address is what ultimately routes traffic.

The other main reason for avoiding the public DNS is the suppression of malware. If malware is allowed to intrude, then the system is, de facto, not private. Routing under the DNS will unavoidably attract and allow threats and malware to operate. By routing independent of the DNS, using an encrypted DNS-like central policy engine and IP addresses only, most forms of malware can be eliminated from any system that utilizes the tightly coupled TrustWrx client/proxy/server application layer. The client code can know the IP addresses of the proxy server, and vice versa, and those addresses can be updated by the server at will. Without a DNS-based domain name, threats and other message-based malware cannot be sent to a message or web address, or to an IoT device. TrustWrx messaging is built entirely on IP address routing – through a central policy engine - without the use of the public DNS.

- **Fallacy No. 11 - "Security should be baked into the application layer"**

This is heard often as it seems to make sense that since effective security has been so elusive, and the application layers are extremely vulnerable, the app layer is where security should be fixed. There many application code vulnerabilities that expose apps to threats. Common attacks at the application layer include; Cross Site Scripting, SQL Injection, LDAP Injection, Cross Site Request Forgery and Insecure Cryptography Storage.

While rigorous and safe coding practices can protect against these threats, most other major threats attack upstream from the code; at ports, in the OS administrative layer and other areas from which the application code is separated. Trying to battle those threats in the static application layer would be futile as production application code could not possibly watch and adapt to the rapidly evolving threat space, or manage other weak points where low-level threats do their damage.