Best Practices in Computer Network Defense: Incident Detection and Response
M.E. Hathaway (Ed.)
IOS Press, 2014
© 2014 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-61499-372-8-54

Back to Basics: Beyond Network Hygiene

FELIX ‘FX’ LINDNER[a] and SANDRO GAYCKEN [b]
[a] *Recurity Labs GmbH, Germany*
[b] *Freie University of Berlin, Germany*

Abstract: In the past, Computer Network Defense (CND) intended to be minimally intrusive to the other requirements of IT development, business, and operations. This paper outlines how different security paradigms have failed to become effective defense approaches, and what the root cause of the current situation is. Based on these observations, a different point of view is proposed: acknowledging the inherent composite nature of computer systems and software. Considering the problem space from the composite point of view, the paper offers ways to leverage composition for security, and concludes with a list of recommendations.

Keywords. Building blocks, cyber defense, cyber security, decentralized public key infrastructure, full stack security, Harvard architecture, information flow control, LangSec, micro kernels, moving targets, separation kernels, verification process.

Introduction

Defending computer networks can appear to be an always losing position in the 21st century. It is increasingly obvious that the state of the art in Computer Network Defense (CND) is over a decade behind its counterpart Computer Network Offense (CNO). Even intelligence and military organizations, considered to be best positioned to defend their own infrastructures, struggle to keep the constant onslaught of attackers with varying motives, skills, and resources at bay. Many NATO member states leave the impression that they have all but given up when it comes to recommending effective defense strategies to the entities operating their critical national infrastructure and to the business sector.

At the core of the problem lies a simple but hard historic truth: currently, nobody can purchase secure computer hardware or software. Since the early days of commercial computer use, computer products, including the less obvious elements of the network infrastructure that enable modern use of interconnected machines, have come with absolutely no warranty. They do not even promise any enforceable fitness for a particular purpose. Computer users have become used to the status quo and many do not even question this crucial situation anymore.

The complete lack of product liability was and is one of the driving factors of the IT industry as it fosters a continuous update and upgrade cycle, driving revenue. Therefore, no national economy that has any computer or software industry to speak of can afford to change the product liability status quo. Such a change would most likely exterminate a nation’s entire IT sector immediately, either by exodus or indemnity claims. The same economic factor caused the IT industry to focus research and development efforts on functionality aspects of their products, adding more and more features, in order to support the sales of the next version of products. Simply put, there is no incentive to build secure and robust software, so nobody does it.

Over time, we have built an IT landscape which consists of many rotten building blocks. Gerald M. Weinberg’s Second Law is often quoted: ‘If builders built buildings the way programmers write programs, then the first woodpecker that came along would destroy civilization.’[1] When it comes to CND, this situation is aggravated by the fact that so-called security software—the very building blocks that we try to use for our defenses—are, by far, of worse quality than anything else.[2] Statistically, not actually using it would be more secure.

This paper will explore what does and does not work in defense, and discuss how we can reduce the defense problem to a building block problem.

1. Composition – Why Basics Matter Most

Computer systems are, like many other things in engineering, constructed by composition. The same actually holds true for attacks (so-called exploits), which are composed of software flaws and incorrect functioning of pieces on the victim’s side. This creates a Weird Machine[3] on the victim’s system that allows the attacker to do what he wants.

From a security point of view, the composition of computer systems is a crucial feature. It reaffirms that any IT-system is, in fact, not just one system, but a heterogeneous multitude of systems, with many different facets and properties, a variety of relations and entanglements with one another, and – by now – in constant flux because of continuous updates, structural changes, addendums, and other practices. All these aspects are relevant for security. Put together, they create a formidable problem.

Unfortunately, the composition aspect was, and often still is, ignored by defense approaches. Some of the more common defense approaches will be reviewed in the following sections.

1.1. The Perimeter Security Paradigm

At the end of the 20th century, computer security issues were still considered more of an organizational problem than a fundamental technical one. In hindsight, this was probably more true so a couple of decades ago, when systems were significantly less complex and their building blocks smaller. The dominating principle of UNIX was one of programs that ‘do one thing, and do it right.’ So the general assumption was that a competent operator of a system could also properly defend it.

The Perimeter Security Paradigm simply describes an organizational approach to limit the exposure of not-so-well administrated machines towards a potentially hostile network. With that in mind, the idea of a handful of firewalls protecting the network was born. On the ‘inside,’ the fragile building blocks could continue to be used (and more could be added), while the ‘outside’ had to be prevented from affecting them. In addition, many organizations retreated to the high ground argument that their network is not connected, and hence not exposed, to any hostile network.

This idea is, however, antithetical to the value of having networked computers in the first place. In order to reap the benefits of communication, one must be able to communicate. Accordingly, more and more interconnections were added and the perimeter simply vanished over the years. Recent trends like Bring Your Own Device (BYOD) are only the final nail in the coffin.

By now, it is clear that the Perimeter Security Paradigm has only delayed a broader recognition of just how vulnerable the building blocks are.

1.2. The Selected Vector Security Paradigm

In a quite similar fashion, computer security developed other focal points at which to implement security. The guiding principle in this case was usually one of minimally invasive measures, applied *ad hoc* to a specific system along with other needs. These measures were applied in a surgical manner to only a very few places, which had been the most common vectors of attacks, so no other 'critical' specified function was disturbed.

This approach, however, is dangerously flawed from the outset. Attackers are not like natural catastrophes. They can analyze their targets for vulnerable elements. Isolating single, selected vectors only shifts them onto a different, less observed, and less protected vector.

An example is encryption. Even today lay people and encryption technology salesmen tend to think that encryption can solve everything—if simply everything could be encrypted, everything would be safe. This reasoning, again, pays little heed to the composite nature of information technology. Encryption can only protect certain content under certain conditions. It will not protect the operating system in charge of the encryption process and in charge of holding the keys. Thus, selling encryption as a critical guard at a critical gate for overall system protection is clearly mistaken, just like any other kind of protection focused on selected vectors. In a composite system, there is no critical gate: everything is a gate.

1.3. The Detection Paradigm

The Detection Paradigm was the next step in defense approaches that completely ignored the composite nature of both computing environments and attacks. Under this paradigm, we subsume all approaches that try to detect attacks, be they anti-virus software, intrusion detection systems, or its more recent sister—intrusion prevention systems.

The basic idea is to detect malicious behavior on the computer system or the network *while* it is occurring. Besides the inherent and well-known flaw in this approach, namely that the malicious behavior must be more or less well known before it can be detected, this approach also fails spectacularly at recognizing the composite nature of attacks. Exploitation frameworks easily demonstrate this,[4] where the building blocks of the attack are composed individually for each attack. Not surprisingly, individually composed attacks are rarely detected by the systems that are deployed for this exact purpose.

On top of that, the situational awareness provided by those detection systems is a worst-case scenario for the defender. If the attack is obvious enough for the system to detect it, it could also be prevented upfront, so no additional benefit whatsoever is achieved. If the attack is only an indicator of something larger going on, the defender is pushed into a real time verification requirement in order to still have a chance to react. Even the later addition of information correlation using Security Information and Event Management (SIEM) solutions, aiming at improving situational awareness, cannot change this underlying race condition that is almost always a guaranteed losing point for defense.

1.4. The Vulnerability Identification Paradigm

The last of the four paradigms is the idea of attack prevention by reducing the number of known vulnerabilities in computer systems. This approach is about as old as the Perimeter Security Paradigm, but at least it begins to acknowledge that the root cause is flaws already present in computer systems. However, this approach also falls short of taking the composite nature of these systems into account.

Due to the complete lack of legal enforcement measures to compel software vendors to produce more robust and hence more secure software, the Full Disclosure movement was developed. System administrators, software users, and security enthusiasts joined this movement, and started to publicly report identified security vulnerabilities, thereby shaming vendors into fixing them. The movement managed to achieve part of its goal, depending on the respective vendor. Most countries now have both Computer Emergency Response Teams (CERTs) that track the vulnerability information published, and openly accessible databases like the National Vulnerability Database[5] that maintain catalogues of them.

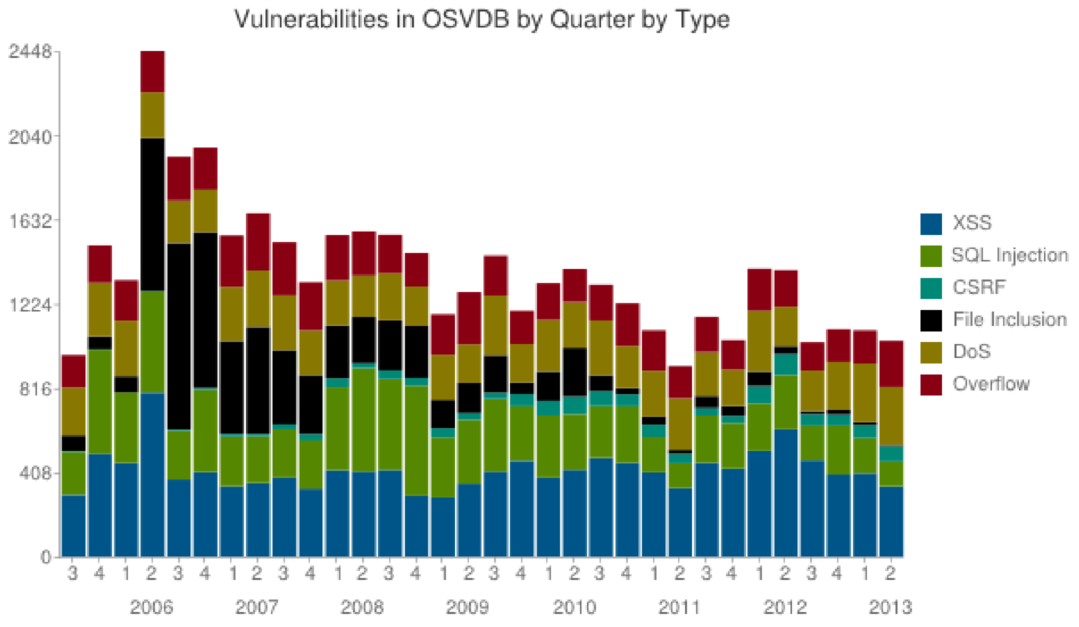

Figure 1: Vulnerabilities in OSCDB by Quarter by Type

Full Disclosure, however, only works as long as the information is coming in. This was the case for a long time, since there was no other legitimate (legal) use for information about security vulnerabilities, other than the modest fame connected with having discovered them, so they were openly shared, but the global rise of nation state CNO operations has created a lucrative market for turning exactly this type of information into attacks, with prices up to $250,000 per item.

While the total number of vulnerabilities reported has been relatively stable over the last decade, our recent research has shown a sharp decline in reports of the type of vulnerabilities considered useful for CNO—namely weaknesses in server software as well as in commonly used desktop client programs, like web browsers. Regrettably, even the overall number of vulnerabilities reported appears to be in moderate decline, which does not correspond to the amount of vulnerabilities actually discovered.

Full Disclosure also drives the development of security patches by the software vendors, at no profit for them. Therefore, reduced public reporting of security vulnerabilities is economically in the interest of the software vendors. Perversely, it is also in the interest of many operating entities, because if vendors do not issue any patches, no systems needs to be updated, and operating costs go down. There is an already discernible push from some software vendors to regulate security vulnerability information, because of the described economic benefits.

The Vulnerability Identification Paradigm, as well as the Detection Paradigm, however, depends completely on the availability of information. Products like vulnerability scanners that inspect computers on the network for known weaknesses are already losing efficiency due to the lack of detail in publicly accessible information. This trend will continue as incentives to keep information under wraps increase, and this defense paradigm is expected to soon lose much of its significance for network security.

Additionally, the Vulnerability Identification Paradigm’s failure to address the composite nature of systems often leads to unaddressed issues such as the fact that the vulnerability stems from an interaction between components of a system, especially when these come from different sources. It is easy to see how multiple parties will try to blame everyone but themselves. Another problem is to know how and by whom a component will be used, and what assumptions a user will have.

2. Leveraging Composition for Defense

As outlined above, the composite nature of attacks as well as of the computer systems to be defended should be the focus of any long term future attempts to improve defense posture. The following recommendations offer realistic, economically feasible, and effective approaches to network security.

Although these recommendations are organized from the most general to the most particular, they should be considered as a whole. It also has to be pointed out that they refer to systems built in the future and are not meant to be retrofitted into legacy systems.

2.1. The Prevention-Detection-Recovery Triangle

Efforts in CND have historically been entirely prevention-centric. As understandable as this focus is, it just does not represent the real world. There is no perfect prevention, neither in the fifth nor in any other domain. Future developments need to take into account from the start that attacks will continue to evolve faster than defense can keep pace with, and that current as well as evolved attacks will be successful from time to time.

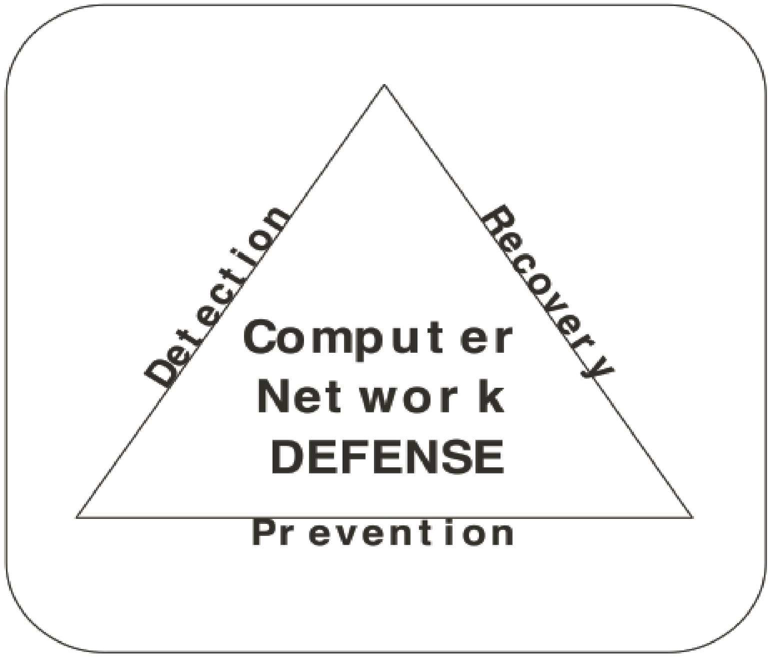

Figure 2. CND Triangle: Prevention, Detection, and Recovery.

Looking at defense in a more realistic manner should also be about detection. This detection, however, has very little in common with what was described above as the Detection Paradigm. It centers on the idea that an attacker loses his advantages the moment the attack is successful. This is also often referred to as the 'defender's home field advantage.' Instead of trying to detect attacks while they are in progress, which has proven not to be very successful, the focus should shift to detecting the intruder once the intrusion has taken place. Post exploitation activity is a relatively long process when concerning valuable targets.[6] New detection approaches should take into account that illegitimate use of computer resources—whether by an insider or an external threat that successfully elevated its privileges into a system—is when the *actual intrusions* can be best identified.

Considering that complete prevention is not possible, and that the capability to detect misuse is independent of concepts like 'inside' and 'outside' threat, leads us to a third element of CND: recovery. Compromised resources can no longer be considered trustworthy in any way. However, in contemporary wisdom, trying to remove the artifacts left by an attack can solve this issue. This practice, however, is rarely successful and only aids future intrusions, because the weakness initially used for the attack is not identified and hence not remedied. The reason for this questionable practice is that today's systems cannot be recovered to their state before an attack due to their sheer size. Reinstalling a single computer after a virus infection is easy, but the same is simply impractical for an Enterprise Resource Planning system (e.g. SAP ERP) or an Industrial Control System (ICS).

Therefore, our first recommendation is to **significantly increase the granularity of the building blocks, making the individual building block significantly smaller** than what is done today.

Interestingly enough, the biggest single building block today, commonly referred to as the Cloud, already makes use of this small building block concept on the resource layer. Individual host machines as well as virtual machines running on them are small blocks with which actual applications, databases, and distributed storage are built. Detection and especially recovery are greatly improved in the Cloud. The virtual machines can be silently inspected from the hypervisor side and be transparently removed and replaced in case of security concerns.

2.2. Speculative Clean Slate

In another twist of 'back to basics', revolutionary ideas for computer security developed in the past decades should be considered. Many of these ideas would have brought IT-security from its immature and dangerous state to one with satisfactory security and higher standards, but they were never very popular. A major problem was that they were often more costly or less efficient and thus provided no incentive for heightened security, or they did not fit well into the existing legacy of technologies. Many of these ideas, in fact, proposed a 'clean slate' as a condition. Considering that, in the past, security concerns were low to nil, there was no real reason to start any reform of the existing environment despite the fact that virtually everyone knew how vulnerable it was and that the decisions taken were irresponsible. Politics could have enforced these more innovative ideas, but politicians were never tech-savvy enough to make controversial decisions against a larger market mainstream. Therefore, many of the very good ideas about IT-security remained speculative. But this is not to say that they are unrealistic. First, in times of a drastically changing cyber security environment, the top shelf ideas should be revisited. Second, many of these ideas had their own little microevolutions in the computer sciences and have often reached levels of maturity sufficient to be implemented in present-day machinery, without any, or at least without significant, loss in performance. Exploring these ideas more thoroughly should be especially interesting for cyber defense.

These are some examples of these innovative ideas that should be considered:[7]

- Harvard architectures: Basic architectures today are 'von Neumann' architectures, which do not distinguish between data and programs. A switch to 'Harvard' architectures would change this, thus making it much harder for an attacker to redirect code flow into data.
- Moving Target architectures: Architectures could be designed to move around in their configurations, thus confusing attackers and rendering it more likely that their attacks generate problems or detectable patterns, and are discovered.
- Microkernels: These operating systems function with much less code, rendering attacks on the OS-level much harder. Some of these are already at work in aviation.
- Formal specification and verification: Well-understood processes could be better specified, in a formal way, and applications and operating systems of smaller code basis could be verified. This way, processes, applications, and operating systems would be mathematically proven to be correct and without typing errors or buffer overflows and similar faults.
- Separation kernels: These operating systems have functional separations within the kernel, rendering a migration of attacks much harder.
- Information Flow Control: This technique could recognize and disrupt illegitimate data flows.

- Full Stack Security: As a part of the clean slate paradigm, the composite nature of IT-systems should be addressed as well. In all critical systems with serious and resourceful attackers, securing all components is unavoidable in the long run. The whole system and the full stack must be secured. Any vulnerable layer could be used as an access vector onto some or all functions of the system.

Many of these ideas had very interesting approaches and prototypes over the past decades, and have been formulated (though never implemented) as requirements. An example is the orange book, an old standard for the evaluation of computer security which – sadly and strangely – had much tougher views on security in the 80s than today.[8] They must be revisited these days.

Militaries will also have to seriously consider changing some of their present paradigms back to older ones. High Security-IT simply does not work very well in an environment constituted by 'Network-Centric' and by 'Responsibility to Share'. This, to quote a frequent phrase of sci-fi robots (which mostly blow up right after saying it), 'does not compute.'[9] Disconnecting the networks and switching back to more swarm-like, decentralized tactics and strategies, based on carefully engineered versions of 'commander's intent,' is a clear task ahead.

This is not necessarily 'retro.' It invites a continuous use of IT and high tech, only of a different kind and in an entirely different overall structure. We should not forget that those military paradigms have been developed for a reason. The 'Network-Centric' paradigm was invented to allow coordination of a diversity of troops against highly flexible and mobile adversaries. The same could be done with swarm intelligence models, but without the vulnerabilities caused by 'network' and by 'centric' (if done properly!). The 'Responsibility to Share' paradigm was introduced to enable militaries to cope with the overwhelming amount of information available, which they automatically gather in these times of vastly more efficient (and numerous) intelligence and analysis methods. A 'many eyes' principle, having many people look at everything, seems an obvious choice to confront this problem. Yet in a digital environment with uncertain security features everywhere, 'many eyes' almost always include hostile eyes. Recent events have demonstrated as much. But again, the same functionality could be provided by a smarter use of technology. One idea would be to relocate some of the NSA's capabilities in semantic web analysis onto their own semantic web, enabling a digital process of 'many eyes' and, if done properly, disabling malicious and unauthorized users; an insecure semantic web analysis would be a grave single point of failure.

2.3. Protecting Interfaces Using LangSec

The smaller the building blocks are, the more communication is required between them. This is a desirable outcome, since communication interfaces are where security can be most easily modeled, implemented, and enforced. The LangSec movement[10] is a language-theoretic approach to achieve that.

Handling the composition of computing systems is arguably the hardest task of both security theory and practice. A system composed of parts with well-understood properties typically has emergent properties that are hard to derive, validate, or even detect from the properties of the parts. These new properties often come as a nasty surprise, creating vulnerabilities that only manifest when building blocks are combined.

The language-theoretic view of security examines system and program components as computational automata, both in isolation and when combined into larger systems. This approach has led to the discovery of serious vulnerabilities in the X.509 PKI infrastructure, remote physical layer frame injection in 802.11b and other wireless protocols, and attacker-driven computation in the binary programs. Defensively, it also points out the way to better implementing security through message validation, and the conceptual separation of code between input recognition and processing. This field explores how to employ language-theoretic principles to construct software that are robust by design and expose as little state and computational power as possible to adversaries.

The idea is to find a 'sweet spot' between formal software validation and the collective experience of both software exploiters and defenders in the field. Language-theoretic security offers a way to design protocols and build systems that can actually be validated and avoid large classes of bugs. Various success stories in both attack and defense have shown the efficiency of this theory in direct practical application.

While the approach initially sounds theoretical and over-formalized, it is actually of very practical nature. Consider the example of a larger system development project with multiple parties. It is common for communication interfaces to suffer from different interpretations of messages sent between them. This is also where most attacks will bring their pressure to bear, since any misinterpretation can almost always be leveraged for an attack. With the LangSec approach, the parties to a communication will specify simple lists of what the content of the messages will be. Once all parties agree on these requirements, a tool will determine the minimal language complexity class[11] and an appropriate formal grammar, and then generate the program code for all sides involved. The formal grammar can later be used to independently and automatically test whether the integrated version of the program code still fully complies with the specification. Therefore, the approach produces secure and verifiable interfaces while reducing development cost and time.

Another use of LangSec is the creation of normalizers,[12] which reduce language complexity and enforce formal grammar on incoming data in order to protect legacy fragile building blocks that consume their output.

2.4. Decentralized and Fine Granular Trust

Computer Network Defense is not an end in itself. The goal is to obtain and maintain control over functionality and data. However, even with perfectly verified and working systems and networks, it is still of paramount importance to handle identification of people as well as authorization of their activities. This problem space has recently experienced a sharp increase in attention due to insider threats and leaking of classified data.

Authentication and handling of cryptographic key material remains challenging. The case of the Dutch certificate authority 'DigiNotar'[13] has once again demonstrated that hierarchical approaches are too fragile and easy to attack, since the adversary immediately gains control over everything below his intrusion point in the hierarchy, and detection by the affected entities is close to impossible. However, alternative decentralized approaches like ISO 20828[14] have been specified and practical implementations of systems derived from those approaches are under active development.

Agile, decentralized, public key infrastructures (PKI) separate the authentication from the authorization problem, eliminate practical issues like certificate revocation, and provide a graceful migration path from centralized hierarchical infrastructure.

Recommendations and Conclusions

The following steps should be undertaken to reduce risks to the building blocks of our network defense:

1) Persuade political decision-makers in a post-Snowden era to act to guide good policies, programs, and standards along the lines of the LangSec and dispersed PKI recommendations of this paper.
   This must be the first and most important step. In recent years, militaries have shied away from taking this step, as they were aware of the multitude of political problems that it would entail for them. Politicians will ask why the military made poor decisions in favor of insecure IT, and why they did not do a proper job to protect their infrastructure. Moreover, politicians will not allocate new money, so militaries may have to solve expensive IT-security problems by getting rid of tank battalions. Neither is it very popular, given the current political and economic climate in security, so militaries tended to pretend that all was fine, while trying to change slowly and 'under the radar' with only small and careful demands for increases in their IT-security budgets. Militaries have been frequently thankful for convenient and convincing lies from the IT-security industry. But politicians need to know that their high-tech, network-centric, all-sharing military apparatus is simply not operational, as soon as their adversaries are no longer goatherds with Kalashnikovs, but a determined high-tech military with a functioning secret service.
2) Acquire and keep an appropriate workforce and R&D-capabilities able to evaluate, monitor, implement, and defend critical infrastructures using a LangSec approach and PKI.
   Experience has shown that this demand is far from trivial. Militaries with a lot of moving personnel, low pay, and few incentives for a high-end professional IT-security workforce tend to fail to get the personnel they need. Bad security personnel make bad security choices and do not understand risks, demands, and options.
3) Identify and formulate high security IT demands, notwithstanding market pressures, legacies or conventional wisdom.
4) Incentivize a high security IT market through military R&D contracts and acquisition specifications.
5) Punish producers of inadequate products and markets with liability fines, business license suspensions, and penalties for downstream losses and by favoring high security products in acquisition cycles.
6) Use modernization cycles in militaries and the economy to move away from insecure solutions and onto high security IT.

In conclusion, we have argued that small building blocks and high security IT are feasible and ever more necessary paradigms for secure information societies. These concepts can secure our IT-environments to a degree far above current standards. Much of it can be composed using communication protocols automatically derived from formal grammar, and can be authenticated using decentralized public key infrastructures.

The underlying issue of a bad market without alternatives or product liability can be mitigated by nation states and militaries acquiring new IT systems, with properties like the ones described here as required elements of new projects' specification. Only then will the incentive to build secure, defendable, and recoverable building blocks outweigh the economic benefits of making more of the same un-defendable IT systems that we continue to spectacularly fail to protect today, but that we all still depend on.

References

[1] Chemuturi, M., 2010. Mastering Software Quality Assurance: Best Practices, Tools and Technique for Software Developers. Fort Lauderdale: J. Ross Publishing, p. 9.

[2] Veracode, 2011. State of Software Security Report: The Intractable Problem of Insecure Software. [pdf] Burlington, MA: Veracode. Available at: <https://media.blackhat.com/bh-eu-12/Wysopal/bh-eu-12-Wysopal-State_of_Software_Security-WP.pdf> [Accessed 5 November 2013].

[3] Bratus, S., Locasto, M.E., Patterson, M. L., Sassaman, L., and Shubina, A., 2011. Exploit Programming: From Buffer Overflows to 'Weird Machines' and Theory of Computation. [pdf] Available at: <http://www.cs.dartmouth.edu/~sergey/langsec/papers/Bratus.pdf> [Accessed 5 November 2013].

[4] Miller, M. (Skape), 2004. Metasploit's Meterpreter. [pdf] Available at: <http://dev.metasploit.com/documents/meterpreter.pdf> [Accessed 5 November 2013].

[5] US National Vulnerability Database. Available at: <https://nvd.nist.gov/> [Accessed 5 November 2013].

[6] It is instantaneous for attacks on individual computer users at home.

[7] Gaycken, S., and Austin, G., (in press). Make Highly Secure Computing the Dominant Paradigm for International Cybersecurity. EastWest Institute Policy Papers.

[8] US Department of Defense, 1985. Department of Defense Trusted Computer System Evaluation Criteria. Available at: <http://csrc.nist.gov/publications/history/dod85.pdf> [Accessed 5 November 2013].

[9] Lost In Space, 1965. [Film] Directed by Irwin Allen. USA: Fox Television Studios.

[10] LANGSEC: Language-theoretic Security 'The View from the Tower of Babel.' [online] Available at: <http://www.langsec.org> [Accessed 13 November 2013].

[11] Wikipedia contributors. 'Chomsky Hierarchy.' Wikipedia, The Free Encyclopedia. [online] Available at: <https://en.wikipedia.org/wiki/Chomsky_hierarchy> [Accessed 5 November 2013].

[12] Lindner, F., 2010. Preventing Adobe Flash Exploitation. [pdf] Recurity Labs GmbH. Available at: <http://recurity-labs.com/content/pub/Recurity_Labs_Whitepaper_Blitzableiter.pdf> [Accessed 5 November 2013].

[13] Wikipedia contributors. 'DigiNotar.' Wikipedia, The Free Encyclopedia. [online] Available at: <https://en.wikipedia.org/wiki/Diginotar> [Accessed 5 November 2013].

[14] American National Standards Institute, 2006. ISO 20828:2006. [online] Available at: <http://webstore.ansi.org/RecordDetail.aspx?sku=ISO+20828%3a2006> [Accessed 5 November 2013].