Securing the Future Software Enterprise

Abstract

The future of the enterprise can be secured provided that it is properly organized and operated with full understanding of economics. The current concentration on "profit here and now" is extremely harmful to the survival of the economy of the world as a whole and every given enterprise in particular. Why is that? There are two parts to the problem. The first part has to do with the short-sightedness the "money question" forces onto the management and the second part – with the isolation of company parts from each other that causes the requirement that everything brings in profit by itself. Under these conditions the security becomes an unwanted "fifth leg" that brings nothing but unjustifiable costs to the company. However, the situation looks completely different if you take a long-term systemic view of the enterprise. In the long term, we absolutely need security as we need quality and many other things besides money to ensure that the enterprise survives. Once we understand that, we shall realize that we already have the knowledge, technology and tools to actually secure our products and we will apply the research where we see them lacking. And we are going to have a look at tools, approaches, and research once we get the principle problem of understanding the necessity of security out of the way. We will talk about the two main directions the security of the software (and hardware) production can take: "product security" vs. "process security", how they are organized and when they can be best used. We will have a look at the hard problems of security that we face and what research and development is being done to address them. The challenges that a software development company faces are plenty and we will discuss as many of them as the time permits.

-- A presentation at the "IT-sicherheit am Donaustrand", 2014

Introduction

The future of the enterprise can be secured provided that it is properly organized and operated with full understanding of its economics. The current concentration on "profit here and now" is extremely harmful to the survival of the economy of the world as a whole and every given enterprise in particular.

Why is that? There are two parts to the problem. The first part has to do with the short-sightedness of the typical management of the companies and the second part – with the isolation of company parts from each other and the requirement that everything brings profit by itself. Under these conditions the security becomes an unwanted "fifth leg" that brings nothing but unjustifiable costs to the company. However, the situation looks completely different if you take a long-term systemic view of the enterprise.

In the long term, we absolutely need security as we need quality and many other things besides money to ensure that the enterprise survives. Once we understand that, we shall realize that we already have the knowledge, technology and tools to actually secure our products and we will apply the research where we see them lacking. And we are going to have a look at tools and approach, glimpse at the research once we get the principle problem of understanding the necessity of security out of the way.

To illustrate, let’s look at how the simple economic model of the well-known game “Civilization” operates.

“Civilization” is a strategic game with a simplified economic model of cities, countries and the world. In this highly simplified model of the economy, describing the behavior of an entire civilization, the parameter "money" is not the only one that leads to success but rather it is used to serve other areas of society. For example, when you build a library you go to the cashier and convert money to scientific knowledge. The theater is also not built for profit but for spending money on the culture. Almost all of the buildings that do not bear a direct destination "hack loot" represent a direct loss: football stadiums, churches, and tank factories - those just consume money, not make profit, but instead they produce something else: contentment for people, culture, or the tanks.

In principle, you can try to concentrate everything in the world on getting more money - but experienced players will tell you that this option is only meaningful on the finishing spurt - when there is a race to win, when you are actually in the military conditions "it’s either us or them." At other times, you can not ignore any sections of public life - it is necessary to make sure that the culture is taken care of and the science is at a level not far behind (so that foreign tanks don’t overwhelm your chariots), and your production facilities allow you to produce anything you might need, and that the cash account allows to support the whole caboodle.

Once again, it is important to note that most of the objects in Civilization are obviously unprofitable and that's fine - they give non-monetary income and in most cases they determine the success or downfall of the player. You build a theater, a library or a tank, pay for them and don’t complain that they need money. Money is produced by special objects replenishing the treasury - they are important, of course, as an integral part of society but their main role in the game is to support the work of other objects - let

the society work and move forward the progress, culture, carry the flag of the country. Only in a single case it makes sense to be "in the money" - when you want to win politics through buying of votes from neutral city-states. In all other cases, a large cash balance, on the contrary, is rather an indication that you are doing something wrong.

So, why are we talking about that? Money in Civilization is a tool and that what it should be in real life, at least in theory. Therefore, if you have excess money, it is best to invest immediately into something that moves forward some real aspects of life - culture pushes the boundaries of your country, science is discovering all the new electric cars, cavalry and navy are bringing the light of truth to infidels. Since everything around is continually evolving, then the funds should be regularly put into circulation - not in the sense of "revolve in the bank" but through investments in the real sector - because conventional 100 coins in the ancient world is not the same as even in the era of feudalism, even in the absence of inflation. Just to save money has no special meaning - it means that you could invest it in any business but did not - for example, you could mount an expedition to another continent but instead you are wasting away over your gold. Yes, the money can be useful to respond to changes in the situation in a rush - but that usually does not come with a huge effectiveness; for example, you can immediately buy up a bunch of soldiers in the case of the Mongol invasion; but if you act wisely, it is much more effective - including in monetary terms - to prepare them in advance; albeit soldiers are all loss and no profit, yes.

In the real world, it is much more complicated! Yet, somehow it turns out that in a simplified toy world simulator "father of the nation" the different effects of a particular aspect of human activity are taken into account, while in our advanced and such a diverse modern society, it all comes down to one parameter - money. Look at what is happening in the world or in your company - the terms are reductions of this and that, because of the "inefficiency".

In purely totalitarian economies societies somehow engage in culture, science and other things, and only in our purely "liberal" economy and culture, we force the culture, science, and almost the military ... to make money. But, after all, this is nonsense in terms of governance!

There seem to be two important aspects at work:

1) The atomization of society and the economy also applies to the enterprise. In a single society and company things can be divided into "earning" parts and "wasters" of money, as was done in the traditional family - husband works in the field, a wife at home on the farm, and that's fine. Under the conditions of atomization one is forced to survive as best one can. The science and culture in the society and security and quality in the company are forced to earn profits, losing their original essence. Every single part is required to perform, basically, all of the elements of the whole without any regard to its original purpose to survive. The security department now has to “sell” its services, engage in marketing campaigns and calculate its “efficiencies”.

2) Extremely short time horizon has become the norm. Where the top management was supposed to keep a very long-term perspective and support the activities that would cause the company to exist in the distant future, now we are dealing with a non-stop pressure to deliver everything today.

In general, the reduction of all aspects of life and work to make a profit in the monetary sense immediately leads to many fun things.

There are many aspects to our work as a software company producing and selling software products but if we simplify the model we can say that there are a few factors that are involved in long term survival and prosperity of the company. One of the factors is the features of the software. That is your "money production" part; the thing that gets software sold and brings in the money. Too much concentration on this part is dangerous, however.

There are other important parts. We will live aside many of them for the purposes of simplicity. Let's look at the quality. Ensuring the software quality is pure cost, it does not sell as such, it does not bring money. Should we stop spending money on quality? You would be right to assume that we will not. But why? Because the quality of our product influences the future sales, it is not here-and-now but in the future that we will see indirect benefits, often not quantifiable. Still most of us understand that destroying the product quality will lead to deterioration of the market sales, company image, decline of revenues and eventual crumble of the company. So somehow over the years we realized that a completely non-profitable activity is necessary for the enterprise survival.

The same applies to security. Most companies ignore security nowadays. Security is nothing but cost and costs even more than quality. Security is even less visible and its impact is even further in the future. Many managers show short-sightedness and ignore security to concentrate on what brings money in today and tomorrow.

> ***"For most companies it's going to be far cheaper and serve their customers a lot better if they don't do anything [about security bugs] until something happens. You're better off waiting for the market to pressure on you to do it."***
> ***– Brad Arkin, Adobe security head, RSA Conference 2013***

But is that a good idea? Security is like your army in "Civilization" – it is pure cost and you may never actually use it directly but it is a good idea to have it unless you want to see your cities overrun by the American war chariots. Security is a cost that an enterprise must take on to ensure its long-term survival. It is as necessary as other costly things – quality, specialist training, research etc.

So when a company puts the security in a position where the security department has to justify its existence by proving with numbers in hand that they are somehow "profitable" – that's pure lunacy on the part of top management. This concentration on the "money aspect" is going to pay off in the short term but will learn to a crash in the long term. The balance is as essential to a healthy company as it is essential to an empire in the game of "Civilization". One cannot ignore the money aspect and risk running out of money at an unfortunate moment. One cannot concentrate on money and ignore everything else either. We must accept that security is one of the realities of life and it is necessary to have because otherwise "their tanks will crash our chariots".

I hope we are clear on that now.

You may only need a sword once but you must carry it every day.
– Japanese proverb

Product security vs. process security

Now that we understand that security is important and necessary, we can talk about how to secure what we are producing. There are basically three approaches to security in a software enterprise. I use the broad categories of “certification”, “product security” and “process security”.

The first approach is the simplest. You outsource your product security to another company. That external company, usually a security laboratory, will check your product’s security including as many aspects as necessary for a set target level of security assurance and will vouch for your product to your clients. This does not have to be as complicated and formal as the famous Common Criteria certification. This certification may be completely informal but it will provide a level of security assurance to your clients based on the following parameters: in how far the customers trust the lab, what was the target security level set for the audit and how well the product has fared. Some financial institutions will easily recognize the scheme because they often use a trusted security consultancy to look into the security of products supplied to them.

Now, this approach is fine and it allows you to keep the security outside with the specialists. There are of course a few problems with this approach too. Main problems are that it may be very costly, especially when trying to scale up, and it usually does not improve the security inside the company that makes the product.

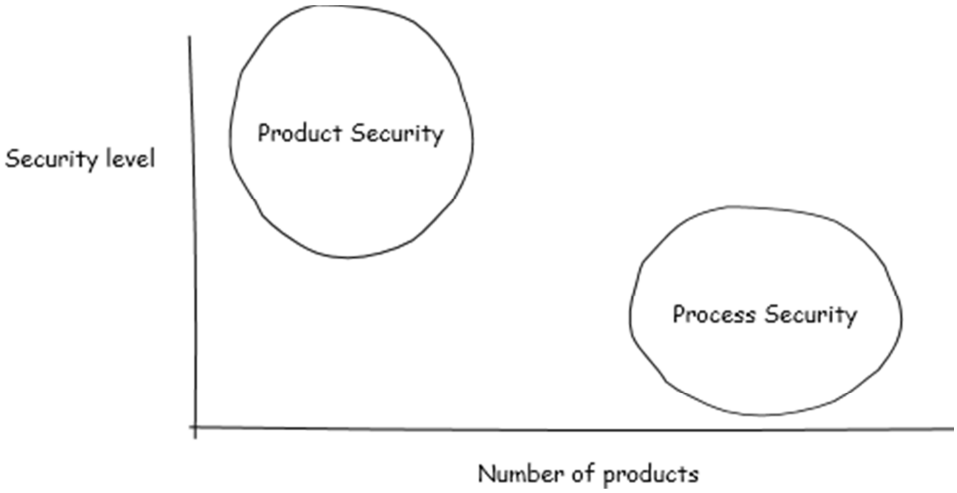
So, if the company desires to build security awareness and plans to provide more than a single secure product, it is recommended that a more in-house security approach is chosen. Again, the actual expertise may come from outside, but the company in the following two approaches actually changes internally to provide a higher degree of security awareness.

One way is to use what I call “product security”. This is when you take a product and try to make it as secure as required without actually looking at the rest of the company. You only change those parts of the production process that directly impact the security and leave alone everything else. This approach is very well described by the “Common Criteria” standard. We usually use the Common Criteria for security evaluations and certifications but this is not required. You may simply use the standard as a guideline to your own implementation of the security in your products according to your own ideas of the level of security you wish to achieve. However, Common Criteria is an excellent guide that builds on the experience of many security professionals and can be safely named the only definitive guide to product security in the current world.

Anyway, in the “product security” approach you will only be changing things that relate directly to the product you are trying to secure. That means that there will be little to no impact on the security of other products but you will have one secure product in the end. Should you wish to make a second secure product, you will apply the same.

Now, of course, if you want to make all products secure it makes sense to apply something else, what I call “process security”. You would go and set up a security program that makes sure that certain processes are correctly executed, certain checks are performed, certain rules are respected and all of that together will give you an increase in security of all of your products across the company. Here we are seeing an orthogonal approach where you will not necessarily reach the required level of security very fast but you will be improving the security of everything gradually and equally.

This “process security” approach is well defined in the OpenSAMM methodology that could be used as a basis for the implementation of security inside the company. Again, OpenSAMM can be used for audits and certifications but you may use it as a guide to your own implementation. Take the parts that you think you need and adapt to your own situation.



The “process security” takes the broad approach and increases the security gradually across the board while the “product security” will deliver you quickly a single secure product with improvements to other products being incidental. A mix of the two is also possible, depending on priorities.

Security by…

Once you decide which approach or a mix thereof you are implementing, you will come eventually to a point where you will realize that you need tools in your development process. And here we come to a yet another interesting discussion of a multitude of possibilities.

You know well that security can be implemented in different ways. For example, there is such a thing as “security by ignorance”. That’s when you never heard of security and don’t know whether your products are secure or not. This is a state of bliss without a worry. Unfortunately, this state usually ends quite abruptly with a security breach and a dose of cold reality.

Since we are talking about security here, I assume that you cannot be in a state of blissful ignorance. You know that security exists and, as we discussed earlier, security matters.

So the next step that companies naturally take is a state of “security by obscurity”, where we try to hide our control mechanisms, important information, interesting data and so on from a casual observer. This is a state where people firmly believe that they made the security good.

One typical example of this is reinventing cryptography. Before I started doing security seriously, I implemented a “cryptographic algorithm” that consisted of taking a user password, reducing it to a single byte and XOR-ing that byte with the stream of data. How much more stupid can you get? I also had a control byte that showed whether the password byte is correct or not, i.e. whether the data decrypted correctly or not. As you can guess, this could be broken by hand in a few minutes. Does this sound ridiculous? Yes, it does. But at the time when I was engaging in this blasphemy I did not know any better, so I thought if I obscured the data so that I cannot read it, it should be ok.

Now, once you get past the stage of “security by obscurity”, you finally make it to a more mature stage where you understand that security is hard and you must take care. At this stage you have a choice of either going the “security by completeness” or “security by isolation” path.

First, let me tell you about “security by isolation”. This is when you try to isolate parts of the system from each other and make them volatile. A great example, easy to understand is the use of virtual machines or cloud instances for sensitive processing. What you do is create a VM with your processing and run the data through it. You also set up some safeguards and detectors to check whether the VM is sound. Should you detect a problem of any kind, you kill the VM and start a new one.

This sounds great in theory but very hard to do in practice. For one thing, there are already viruses that actively target such systems and organize themselves into networks in order to survive the death and resurrection of singular VMs. Think about this too: you set up the processing for a purpose. The purpose is to process some data. You receive the data from somewhere, you process it and you pass it on. You cannot achieve complete volatility, meaning you cannot kill the process together with all of its side effects simply because one of the side effects is producing your data and you need the data. So as long as your VM is doing anything useful at all it is vulnerable.

Anyway, the far more pervasive method nowadays is the “security by completeness” and since we want to get to talking about tools, that’s what we should go on to, right? Security by completeness means that you verify your product to be secure and you make this verification as complete as possible. You basically use all sorts of methods and tools to check whether the product implements what you wanted and behaves according to your expectations under all circumstances.

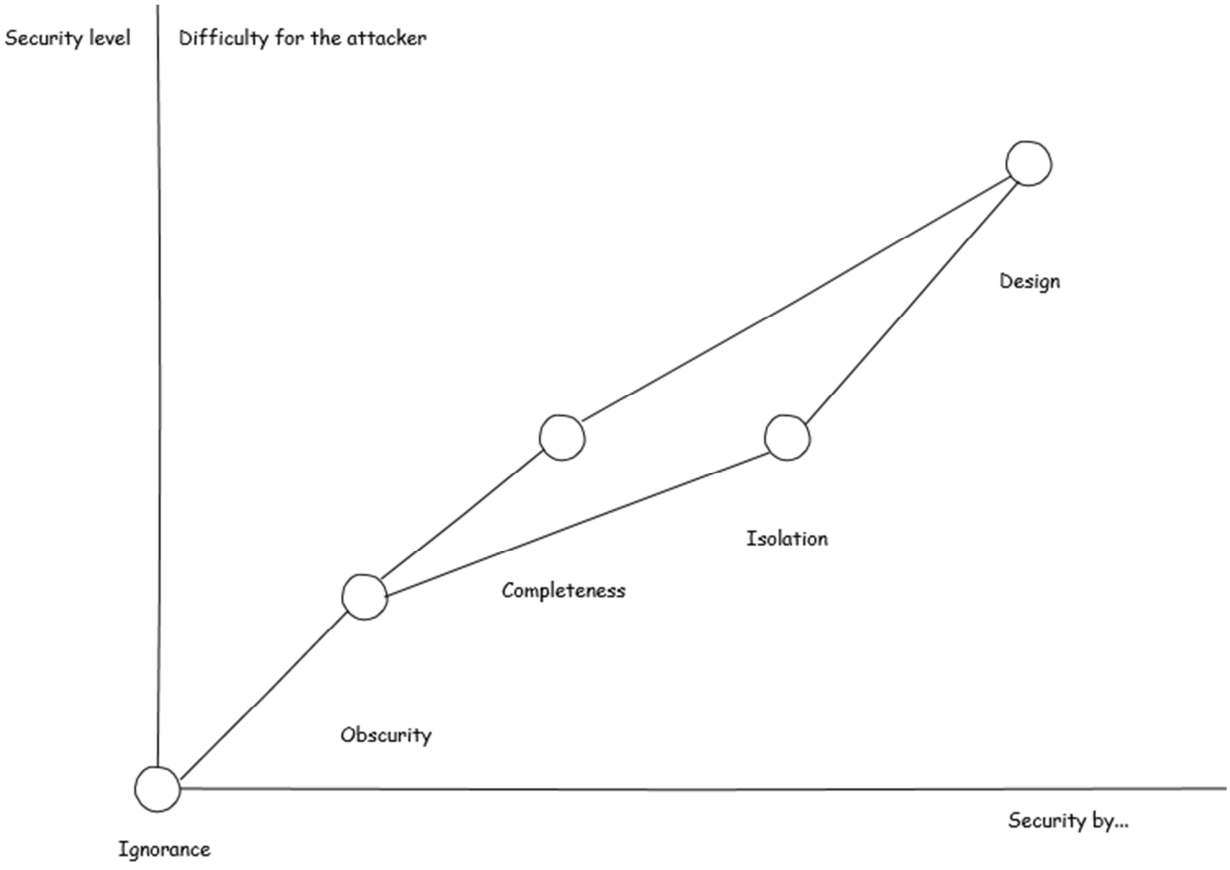
Ideally, you want to verify that the product behaves according to the ideal of behavior that the designer had in mind at all times. Unfortunately, this is very hard to achieve. The translation steps from idea to design, from design to product implementation, from implementation to a running instance take away the information and distort the intent. Still, we can try to do our best in verifying that (a) the product does perform according to our expectations, (b) product does not perform anything unintended and (c) the product cannot be forced into performing out of (a) and (b) bounds even under hostile conditions. That’s where security testing enters.

Basically, the product's behavior according to the specification is covered by QA tests. I usually say that they are a part of the security testing because that's the first part of the security equation: product behaving according to the specification under normal conditions.

The other two tests are usually seen as "real" security testing. Today we have two methods: static code analysis and dynamic code analysis. The static code analysis analyses the code and looks for patterns in data processing that hint at possible security problems. The dynamic analysis looks at the interfaces of a running product and tries to analyze whether some known ways of breaking in could be used on this one. Both methods try to provide as much coverage as possible but on any non-trivial piece of software the coverage is not going to be even close to complete.

And so we fail again, you see. The security by completeness and the security by isolation do not help as much as we would want.

The only way to achieve proper security is to go the path of "security by design" that is often mentioned but rarely implemented. Security by design means that the systems are built to be secure from the beginning. That means also that the systems would not progress in their complexity as fast as they do now. Security costs money and effort, it prevents some of the innovation, it interferes with things. Building a secure system would mean going back to the time of mainframes and continue that evolutionary path, slowly and painfully extending those systems, spending a lot of money on the "fifth leg" and never getting ahead of the game. Who would agree?



So the security is far from perfect. The methods we use do not allow in practice to come close to building a truly secure system. Why do they work? They work because there are other products where the security is ignored or security by obscurity is implemented, so there are much easier targets than any software where security methods are used. That is the whole point, you see. We are trying to make our products harder to break than the next thing around the corner. At its best, we try to make products so hard to break that it costs more than what you could gain by breaking them, effectively making it economically not interesting for the attackers to attack our systems. As long as we can manage to ensure that the easiest way to break in is far more costly than what you gain from breaking in, the security by completeness and security by isolation remain viable methods of ensuring product security. Free market rules.

Security tools

So, that's how we end up where we are with the security industry today. Whether you do product or process security your best bet is to either do the security by isolation or security by completeness. The latter is much better understood and has better tooling while the former is still in its infancy. So, for production purposes and unless you have too much resource you will likely stick to the security by completeness and hence will use all the same traditional tools that the rest of the industry does. Welcome to the state of the industry, so to speak.

What is the state of the industry then?

What do we need to cover in order to make sure the security assessment is complete in simple terms?

We need to look at the development environment, whether it is secure, and this part has been well taken care of by the IT departments, who do a surprisingly good job nowadays. You could say if you have a decent IT department, you do not need to worry about your network security.

We must look at the security of our suppliers and the tools and libraries they supply to us, the so-called 3rd party security. This part is currently not so much developed. There are efforts underway by various companies willing to sell to us various tools with which we can assess the security of the supplied software. Mostly though, this is about libraries – tools like Veracode and Palamida come to mind. They are very useful and they are a good start but they do not cover all our needs even for libraries. As for the tools, compilers, IDEs, version control systems and so on, we have no tools. The best we can hope for is that if they were tampered with someone will notice eventually. Moreover, the only time you actually do a security audit of your supplier would be during a Common Criteria and that is not likely to happen for many of the companies.

One place where we seem to have a lot of tools and techniques is the security testing. Now, there are basically two types of automated security testing that you can apply: the static code scanning and the dynamic code scanning.

The static code scanning can be done with a variety of tools. There are free tools from the Open Source community, there are tools from specialized vendors like Checkmarx, there are static code analysis services from the cloud like in the case of Veracode. There are two important things to note here. One

important thing is that even if you use the simplest of the free tools – FindBugs – to scan your software and fix all of the reported issues, the security of your code increases a lot. This is our internal observation: teams that pay attention to the scanners' results and fix them timely consistently show much better results in security audits than others. So the point is not simply to run the tools but to make sure the developers pay attention to the code and think carefully about things they develop. The other important observation is that static code analysis covers at best 10-20% of the known security problem types. If you get more of them, they will all find more or less similar problems, just more or less of them with lower or higher false positive rates. But you will not be able to get anywhere covering the whole of the CWE database with those scanners.

The dynamic scanning tools are also available both from the Open Source community and specialized vendors. There are vendors that also take the tools into the cloud and make then into a service for sale. Dynamic tools operate on the running product so they are much more likely to find real problems. Unfortunately, the coverage is also very low here. We know that the commercial tools cannot find some of the security problems found by our penetration testers. That is just a fact of life – the tools will likely never be as good or as clever as people. Interestingly, the impact of the dynamic scanners is quite different. The dynamic scans uncover more of real problems but since the dynamic scans are positioned as a gate at the integration stage, it is too late to influence the quality of the software anymore. So the dynamic scanners find problems (that we fix, naturally) but they do not improve your development team much. Paradoxical but true.

What are we missing?

We are missing a lot in the 3rd party security. There are no ways to perform audits of the suppliers, sometimes technically, sometimes procedurally. How do you audit a supplier that consists of twenty Open Source developers spread out around the world? We also miss severely in the area of tools. We had to extend Palamida with our own homegrown tools to make it work more effectively based on our private research projects with Fraunhofer Institutes in Germany. It would be great if there was more research into this area for useful methods, techniques, tools.

As I said, static and dynamic code analysis has a lot of room for improvement. Even if it gets twice as good as it is today, it will not be enough, so I think investment into better automated security testing algorithms and tools is very much justified. Think about this: one of the main headaches for anything that can be accessed with a browser is cross-site scripting attacks. Tools for detecting cross-site scripting attacks suck, manual penetration testing can easily uncover most of those and the automated testing uncovers only a small portion – the simplest cases. These tools must get better.

One area where we lack tools, methods and techniques severely is the risk analysis. Today, there are methodologies developed for the risk analysis but there are no tools that would be able to operate on non-trivial products. Performing the risk analysis according to the existing methods manually on non-trivial products is also prohibitively expensive.

The risk analysis is important mainly for the purposes of investment management. We must know where our effort should be invested in the first instance. The management must know where the money should go first and foremost. So we have to pinpoint the places in our security architecture, between our

software components and in the processes where the investment will have the most efficient impact on the risk picture for the company. And this is the kind of thing we are lacking at the moment.

Actually one of our research projects that we are working on right now is the EU sponsored RASEN project that promises to deliver both risk-based testing and test result aggregation into risk analysis. We are very hopeful for this project although the task is really complex and difficult.

There are other tough problems that require security research. As one example is the problem of composition. Quite often you hear people talk about secure components and how they build secure systems because they use secure components.

Secure components are never secure unconditionally. They are what we call conditionally secure. They are secure as long as a certain set of assumptions remains valid. Once an assumption is broken, not met, the component is not any longer secure. Who checks for those assumptions? Who verifies whether the developers upheld all of the assumptions that the developers of underlying components specified? Who checks what assumptions remained undocumented?

When we combine the components together we create a new problem, the problem of composition. This is not an easy problem at all. By having two secure components put together, you don't automatically obtain a secure system. It may well be. Or it may be not.

This problem of secure composition is well known to the developers and auditors of smart cards. And they do not claim to have a solution. And here we are, developers of systems orders of magnitude more complex, dismissing the problem out of our minds like if it's not even worth our consideration. This is a very tough problem and multiple levels and it requires a lot of further research.

Conclusion

Every single website and application, every single computer system gets broken into for fun, money, fame, or accidentally. This is just the way it is and I have to accept this as the current reality.

My message to you is that you have to accept and convince your company that investment in security is a good long-term investment that your company may not survive without. I want you to know that there are methods, techniques and tools available to you once you decide what approach to security you take and the level of security you can achieve today is pretty good. I also want you to remember that it is not sufficient, that we are all hanging there by a thread and we are far from being unconditionally secure. Once you start doing security you should start doing security research. Spending on security is your investment into the future of your company. Spending on security research is your investment into the future of the industry. For the future, we are going to need both.