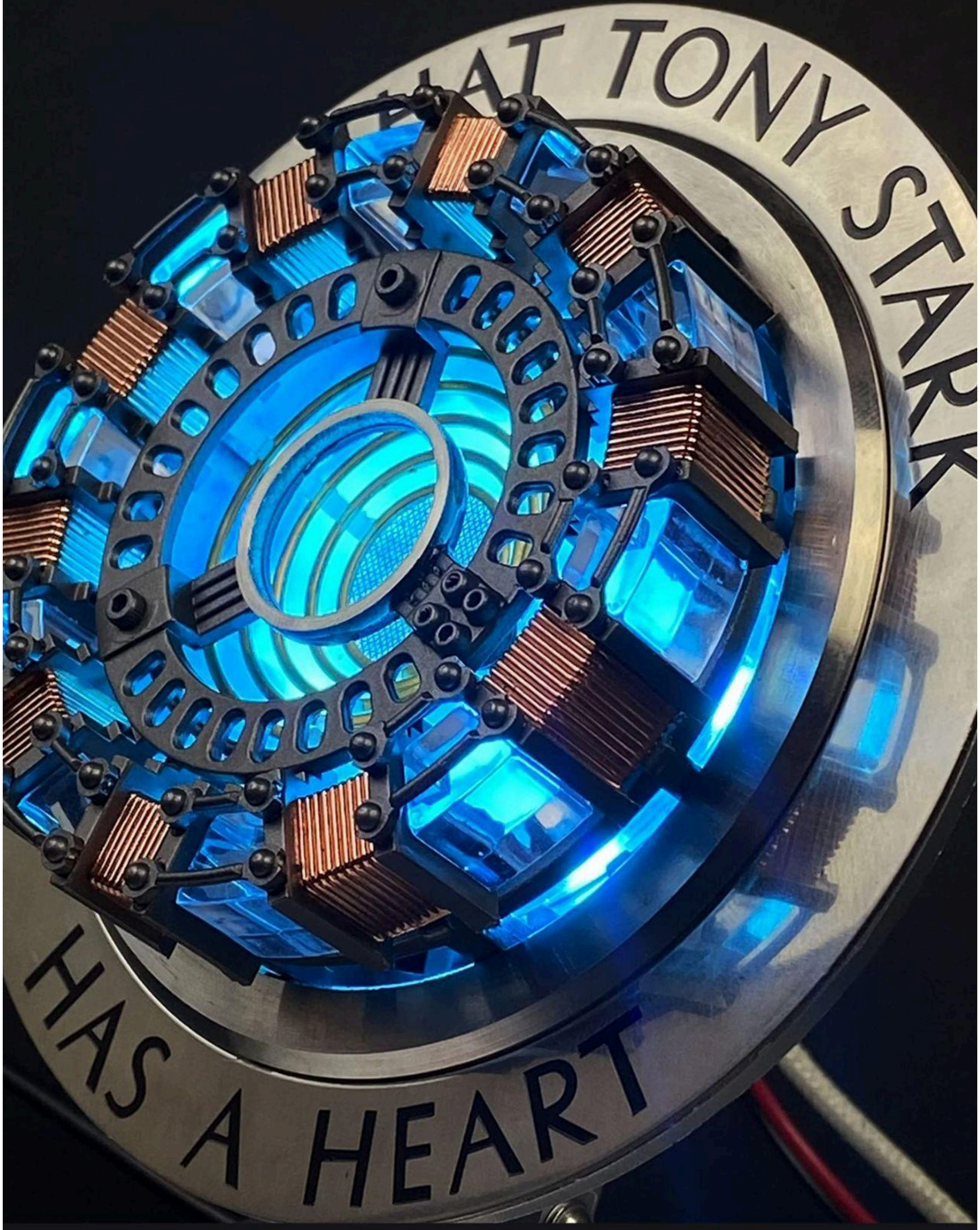
NÃO É SEMPRE QUE VOCÊ PODE ENTRAR NA MENTE DE UM GÊNIO

APPLICATION SECURITY



CÁSSIO BATISTA PEREIRA

LESSONS LEARNED FROM IRON MAN ABOUT SECURE SOFTWARE DEVELOPMENT

AppSec - Iron Man

Application Security through Iron Man's mind

Cássio Batista Pereira

AppSec - Iron Man

Application Security through Iron Man's mind

Cássio Batista Pereira

Independent publishing

Contact:

Cássio Batista Pereira
cassio@cassiobp.com.br

SUMMARY

Preface

Cássio B. Pereira
Application Security Expert

Author

Nice to meet you, I'm Cássio Batista Pereira, a.k.a. @cassiodeveloper, I'm a Software Developer and Architect by training. I work as an Application Security Engineer and help companies and professionals build safer solutions. I have over two decades of experience in the development market in the most varied business segments, where I gained knowledge to work with different technologies, languages, frameworks and processes. I am an evangelist for the Secure Development culture and share knowledge about AppSec, DevSecOps and SDLC, more details here.

You can also find a version of this book here in a presentation format, enjoy.

AppSec - Iron Man

Warning

This book will touch on a few movies of the MCU - Marvel Central Universe, and for that a lot of **spoilers will show up,** all the movies are quite old so, by now, you should have watched all of them. But, in case you are not a fan of superhero movies and haven't watched so far, be warned that there will be a lot of spoilers. So, I either recommend you to watch all the movies before proceeding, or even better, read this book right now, and enjoy the movie experience after (since you will know details that most people have no idea about) even more..

Chapter 0

Brief intro

“Safety is everyone's responsibility”

– Cássio B. Pereira

Tony Stark, also known as Iron Man, is a fictional character in the Marvel Comics universe. He was created by writer and editor Stan Lee, developed by scripter Larry Lieber, and designed by artists Don Heck and Jack Kirby. Tony Stark made his first appearance in "Tales of Suspense" #39 in 1963.

In the Marvel Cinematic Universe (MCU), Tony Stark is portrayed by actor Robert Downey Jr. He is a central figure in the MCU and played a crucial role in the formation of the Avengers.

Background:

Tony Stark is a brilliant industrialist and engineer who inherits Stark Industries, a multinational weapons manufacturer, from his father, Howard Stark. He is known for his exceptional intellect and inventiveness. Stark's life takes a dramatic turn when he is captured by terrorists and forced to build a weapon of mass destruction. Instead, he builds a powered suit of armor to escape, marking the birth of Iron Man.

Iron Man Suit:

The Iron Man suit is a technologically advanced powered exoskeleton that enhances Stark's strength, durability, and abilities. Over time, Stark continually upgrades and refines the suit, creating numerous variations with different capabilities.

Character Development:

Tony Stark undergoes significant character development throughout the comics and MCU. Initially portrayed as a playboy and a self-centered genius, he faces moral dilemmas that lead him to reassess his priorities. Stark evolves into a more responsible and altruistic individual, using his technology to fight against threats to humanity.

Role in the MCU:

In the MCU, Tony Stark plays a pivotal role in the "Iron Man" trilogy, "The Avengers," and several other films. He is a founding member of the Avengers and is instrumental in the battle against various formidable adversaries, including the Chitauri invasion, Ultron, and Thanos.

Legacy:

Tony Stark's character arc reaches a poignant conclusion in "Avengers: Endgame" (2019), where he makes the ultimate sacrifice to save the universe. His legacy lives on, and the impact of his contributions to technology and heroism continues to shape the MCU.

Influence:

Tony Stark's character has become iconic, representing the intersection of intelligence, charisma, and heroism. His journey from a self-centered genius to a selfless hero has resonated with audiences, making Iron Man one of the most beloved and enduring characters in the Marvel universe.

Author's view:

Now that you know (if you didn't before) a bit more about Iron Man and Tony Stark (same person, but while reading this book, you might see both references) let me bring you my personal opinion / view of the character.

First, I know you are anxious about the AppSec and tech part but, it's crucial to understand his mind in order to understand his achievements and remarkable feats, so don't skip this part or later some things will not make sense at all. So, Tony is a human right? Just a man, yes, with a lot of money and I mean, A LOT OF MONEY, a huge knowledge and strong personality, but he is just a man, with problems also, specially in relationships, considering his parents were most of the time absent, he was usually feeling loneliness, that's why (maybe) part of his hobby, was enjoying parties and nightlife spending his fortune with nice girls and awesome cars. But ok, this is just part of his life and character, one important fact for us here is that, he grow as a human being, he learn with his own mistakes, recognize them, and grow, and because of that he wants also everyone around him to grow as well, to improve and to be better, as human beings. And we see all that during his development in the movies, I'm not a big fan of the comics, I might have read one or two when I was teenager, but that's all, my whole focus is on the MCU movies, so please take into consideration that I might miss some information about him.

As I said, as a man Tony knows his limitations and strengths as, I would say, every man should know because this allows and "make easy" our journey called life, and being a genius as Tony is, might sounds arrogant sometimes (all the time), or even careless in some situations, but do not fool yourself, he is not.

In this book, I will take you on a journey through the MCU (with focus on Iron Man), considering the movie trilogy Iron Man 1, 2 and 3 and of course his appearances in other movies like The Avengers, Avengers Age of Ultron, Avengers Infinity War, Avengers End Game, Captain America Civil War, Spiderman Homecoming and maybe others (never remember all), and during this journey we will learn how Tony’s mind works and benefit from it to implement Application Security in our real lives, and it doesn’t matter if you are a Developer, a QA Engineer, a DevOps or a Cyber Security Engineer, if you are involved in the SDL somehow, you must know AppSec. We will see a lot of references to the movies, and learn from these references how Tony thinks and how we should also think to make sure our software is, well, safer in a way.

I strongly recommend you to watch ALL the movies, either during this reading, before the reading, or after reading this book. Things will make more sense, you will have the same emotions and feelings that I had writing this book, and you will have fun and good time watching, maybe you can do a “movie marathon” on a cold weekend alone, or even with your partner, nice time to be together and discover / learn something new. Ah but I don’t like superheroes, Marvel etc… I DON’T CARE. WATCH THEM ALL. You might start liking it, you might see that fantasy and fiction are good to our imagination, see this book!

Chapter 1

Threat Modeling

"The art of finding problems even before the software exists."
– Cássio B. Pereira

Contextualizing

In the first movie Iron Man 1, Tony is kidnaped by terrorists and kept as a hostage in a cave, is the beginning of all for Iron Man to become who he is. And I will not write the whole movie script here, but it's important to set some ground. During the attack of the terrorists on the convoy that Tony is, a few soldiers die and Tony see himself next to an active missile / bomb where is written "Stark Industries", such an irony that his own product now, is threatening his life, and his face just not show surprise that "oh fuck a bomb here…", but is more "oh fuck, MY BOMB here?" because he knew who his customers were, and definitely those terrorists were not on his customer list, so how the hell his missile was there? And this moment is crucial, because he realizes that his father's legacy and now his own work, is being used not to protect America, but is in the hands of bad guys that are killing innocent people and he doesn't want that.

In the cave after a few days, Tony realizes with the help of another hostage that he must escape, because if he creates what the terrorists want America and the world would be even in more danger, considering that they wanted the technology of the Jerico missile, a new powerful weapon designed by Tony. He then pretends to be building the Jerico missile, but he is building something different, something that would be his greatest legacy later on and for that moment, his salvation the Mark I, the first suit, the Iron Man suit, and literally Iron, because that was the materials he had in the cave.

Threat Modeling

It's not clear in the movie of course, but Tony does some kind of Threat Modeling to create the first suit, otherwise he would not create it if he was not sure that it would succeed, or at least a high % chance of success. Because let's remember, he is a man, any strong punch, bullet, knife can kill him, so he did what?

Threats

The main threat was **armed men,** considering few types of guns and machine guns.

Goals

The main goals included:

- To escape from the cave - Go out;
- Go as far as possible - Not stay at the entrance;
- Take no damage - He didn't want to die;
- Attack - In order to avoid some threats;
- Hope to be found by USA;

Requirements

The main security requirements included:

- To escape from the cave - Possibility to break doors and avoid enemies;
- Go as far as possible - Fly;
- Take no damage - Bullet proof;
- Attack - Flamethrower, punches and kicks;
- Hope to be found by USA - Being out of the cave would be easy to the USA Army to find him;

Lessons learned

From this, we can get a few lessons:

- With the material he had, he did the best what he could;
    - Mark I, would not defeat Thanos or even be enough to fight Captain America and the Winter Soldier later, but for THAT MOMENT, was enough.
- Done is better than perfect!

Reflexion

It’s important for us to learn from this first moment of the book asking ourselves some questions:

- How many times have you thought about the threats that your software might face before you release it to production?
- Have you ever considered that your company, brand, software etc might be a target for cyber criminals?
- Did you ever consider the impact on society in case your software is impacted by a cyber attack?
- Did you ever imagine that it’s possible to brainstorm some threats just by the list of functional requirements?

Be honest, if Tony Stark was not 99% sure that the first suit would bring him freedom from the terrorists, do you think that he would do it anyway? Or, he did it because he was sure that he would succeed somehow, considering the threats and the protection that he had, he was basically thinking, the threats are level 3 and I have defense level 5, so let’s go.

In our software development lifecycle, that’s the same mindset we must have:

- What are the threats that the functionality X might face?
- What are the mitigations that I should implement to reduce the impact or even eliminate the threat?
- And do all this, before the software goes to production. Otherwise, we are at risk.

Deep knowledge

Threat Modeling - HOW

I want here to point out some tips for your regarding Threat Modeling:

- Threat modeling should be used in environments where there is significant security risk and not in the whole system.
- Threat modeling can be applied at the component, application, or system level.
  - I recommend the most granular possible (functionality), to don’t have a huge model that will be hard to read and understand.

Use case - Veronica A.K.A. Hulk Buster A.K.A. Mark XLIV

In Avengers Age of Ultron, Hulk got bewitched by Wanda Maximoff the Scarlet Witch, and ran out of control as never before. By then, Bruce Banner was already (almost) able to control Hulk, as we saw in The Avengers and also in the post credit scenes of The Incredible Hulk. But, together Bruce and Tony designed the Veronica in case Hulk gets crazy again, now it would be possible to, at least fight him to control the damages and losses, before was not.

So, Veronica was a 100% Threat Modeling job, it was exclusively designed to fight / contain Hulk in case of an unintentional change of Bruce Banner, and it almost didn't work. During the fight, we can see that Veronica has a lot of functionalities to contain Hulk, not to destroy / kill him, but to stop his attacks, to try to make him sleep with some kind of green smoke poison, to block his punches with a hand locker… and Tony almost lost. Not to mention that in the beginning, some sort of a cage locks Hulk down, unsuccessfully. The movie does not show this, but Bruce and Tony worked together, discussed the Hulk attacks, strengths and weaknesses, possibilities and consequences to create the perfect HulkBuster suit, not only to stop Hulk but also to protect Tony's life. Note that in the image above, as soon as they punch each other at the same time, Tony's face is "surprised", because he knows the power of his Suit, he has all the calculations of how strong he is, and still, Hulk punch was in the same level, in fact the threat is more scary when it happens than when we just think about it.

Threat

The main threat was **Hulk**.

Goals

The main goals included:

- To stop Hulk - Make him stop attacking / damaging innocence, the city etc;
- Go as far as possible - Not stay in the city, but at a non populated place;
- Take soft damage - He knew he would take some punches;
- Attack - In order to archive the first goal as well;

Requirements

The main security requirements included:

- To stop Hulk - Cage, strong and agile enough.
- Go as far as possible - Fly, arm lock;
- Take soft damage - Only iron would not be enough;
- Attack - Fly, arm lock, laser and poison;

Lessons learned

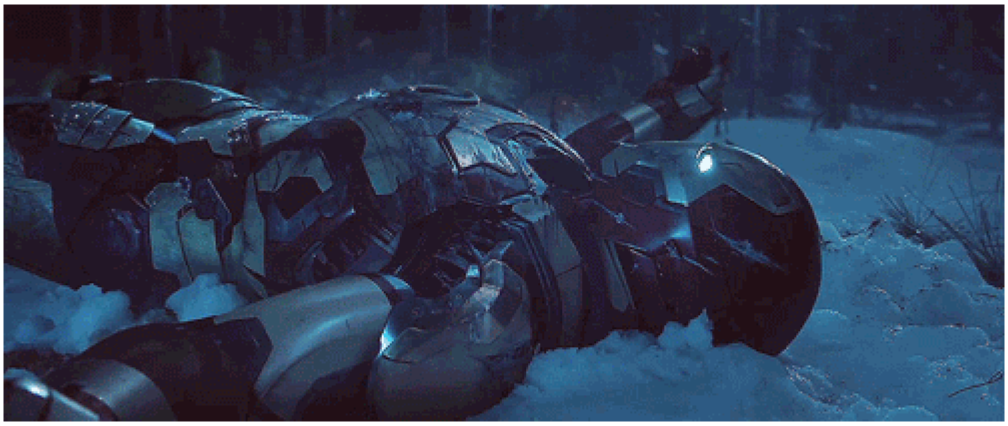
From this, we can get a few lessons:

- The real threat is different than the theory threat, usually worse;
- Even with the whole mitigations in place, the damages still can be harmful;
- Time and research are necessary;
- Communication is very important;
- Be ready to improvise at any time;

Incident Response - Learning from facts and mistakes

Another very effective way of doing Threat Modeling is to use the incidents that happened in the past, history of what and how it happened helps teams to create bullet-proof softwares, at least proof against known threats that were already reality to your context. Like the case below:

In Iron Man III, Tony flew to Tennessee after being attacked by the terrorist group led by The Mandarin, his suit was a prototype that's why the flight happened, before he was just planning the flight with Jarvis. Almost heading to Tennessee and his suit was running out of energy, Jarvis alerted him and he felt almost with 0% energy. Of course he needed to get out of the suit in order to go anywhere, now by walk. But it was winter, he was freezing in the snow. It's good to remind you that, as a man, Tony now was with soft clothes in the hard snowing winter, which is dangerous to any human in these kinds of weather conditions. Also, he was quite damaged after the "fight" with the terrorist group that attacked him, he was hungry, dirty and thirsty and far from basic resources.



And then, when he created the new Spider Man suit later in the Spider Man Homecoming movie, well…. It was equipped with a heater! When Peter fell into a lake and was rescued by Iron Man, Tony said:

It's pretty clear already that Tony's mind is amazing and he is always improving, he had a problem, that he fixed that's pretty basic no? Well, no… our softwares daily presents bugs and vulnerabilities that we don't even bother to check, we just put them into some kind of nice listed name, such as Backlog, and one day if nothing else more important like breathing popup, we will handle it. Tony shows us that we might make mistakes, but we can't keep them, we need to make it better next time. In this case, not only for himself but also for others, we care about others, right? Our software usually intends to bring value to our customer, that intends to bring value to the end customer, but in case of a problem, everyone loses.

Let's also make something clear here, he had a minor problem considering the whole Iron Man threats universe, being exposed to snow is not a "huge bug", but something that considering the environment and circumstances, could kill him, so why not fix it?

Summary so far

Let's have a break from the flow of the topic and summarize some of what we saw so far:

- Your software version 1.0.1 must be better than 1.0.0, better in terms of security, not only new nice features that "bring value to your customer". FUCK IT. If your new feature is a door for a cyber attack, you are not bringing value, you are TAKING VALUE.
- Think like Tony Stark! Ok you might not be a genius, but try it!
- Not only fix bugs, you must improve security. Vulnerabilities and business logic flaws must be fixed.
- There is no security, without proper SDL process and team commitment.
- Imagine that each release of yours is a new Mark suit that Tony built.
- Each suit is designed to protect his life, your software can't just bring value if the value does not protect your customer / user as well. So each version of your software must also protect your customer / business.

Back to Iron Man I

Right after being rescued by Coronel Rodes after escaping from the terrorists, Tony got home and started working on improving already his great creation, the Iron Man suit. He created the first fully operational suit, after a few testing sessions failed (that's what tests are made of, right?) now he was able to have a stable flight, he had the real time monitoring and assistance from the AI Jarvis, and not only iron as material for his suit, but a better and durable material, and of course, an improved version already of the main component of the suit, the power source, The Arch React. Being able to fly, he took his first flight directly reaching the stratosphere and having an icing problem.

At the end of the movie, with Mark III he fixed the problem, now he was able to fly to the stratosphere without icing, but the villain didn't because he was using a version based on Mark II, with the icing problem. Wrong fork dude, wrong fork!

iron man icing problem

When a vulnerability became a feature

1. In Iron Man II at the fight with the Whiplash, Tony suffered a lot of damage from the energy of the Whiplash whips, it was supposed to be an easy fight, but was not the reality. By then, we were already able to see a huge improvement in the Iron Man suit, now Tony didn't need to be at one place to put on (mount) the suit, now the suit was portable in his suitcase, much more effective right?

2. Later in The Avengers, during the fight (misunderstanding) against Thor, he was now able to absorb the energy from the thunder, and get less damage. See? Now his suit was able to take less damage due to a better material, and also absorb the energy from the attacks, he learned that in case some enemy has "an energy weapon" he can benefit from that, isn't that brilliant? The improvements on the suit regarding material are constant, every new version they are better, somehow and this by definition is a continuous improvement process that he has.

3. Then, in Avengers Ultimato his suit now is equipped with so many features that is almost impossible to describe, with the whole nanotech capabilities the limitations were only his creativity I'd say. But, during the confrontation with Thanos, together with Thor and Captain America, he now was able to open his back like a "flower" to be able to absorb the full energy from the thunder that Thor invoked and become a powerful attack weapon firing a strong combination

of Thunder and his Arc React laser against Thanos. Yes, it's fine that Thanos easily handled it, but damn that's genius.

This sequence shows us that his is not only worried about protecting himself with the best material suit, his always improving the capacity of attack as well, and what once was a bug, poor material, a lot of damage and no benefits, now was a suit with a stronger material and energy absorber capable of defending and attacking at the same time. Not to mention some details regarding the energy absorption that the whole suit power source needed to be prepared for, right? How much energy a thunder can produce? How much energy was he able to store before? All this calculations and details are not there, but implicit show us that a fucking hard work was made. Not to mention the portability of the suit itself, from a one place mounting at his garage, to a suitcase and now a kind of a simple "chest badge" that could carry all that with the miracle of nanotechnology.

Now with this in mind, imagine the bugs and vulnerabilities in the software you are working on, worked or will work doesn't matter, they are a pile of Jira tickets in a lost backlog that no one will ever touch? Or worse, they are not even documented any how? Or, you constantly evaluate and test them to make sure that they go through an impact or risk analysis to guarantee a better incident response management when things go wrong? Or even to know what are the consequences of that bug / vulnerability in your environment in case they became an attack?

It's very common to see companies handling their bugs, they have Q&A teams full of Engineers to look for problems in their products and that's really good, they have developers more and more experienced, that tends to avoid many bugs, well, at least the silly ones. It's not so hard to find companies applying techniques like TDD on a daily basis or even Q&A teams that are capable of creating automated tests, this is awesome. It took some time for that to happen, I remember when I started coding when I was 15 around 2002 (last time that Brazil won the World Cup by the writing of this book), the developer was the tester, there were testing teams but not so common like now. Especially because by then, Visual Basic and Deplhi were the peak of programming languages and Web development was still growing and becoming popular. So the Client-Server Desktop systems were much more popular and let's be honest, much easier to test and validate. But, the same does not apply to threats or vulnerabilities.

Bugs are expected, everyone knows about them, accepts them and even enjoys them, like, as a developer I was happy (sometimes) to find a bug and fix it that means I won't "harm" my customer or end user right? Even when the QA Engineer was reporting some stupid bugs, so called non blockers, I was happy it was all part of the improvement of the product (software) to be as perfect as it can be, considering of course, the time and resources for that. But, when you change the word BUG to the word VULNERABILITY the magic happens, no one cares anymore, what I most hear about it is (in order):

1. It's a false positive; (99% of the cases)
2. Our product / company / brand is not a target for hackers (crackers);
3. This will never happen, who would know that?
4. It's very hard for an attacker to do that.
5. Ah, if this happens then we have bigger problems.

With vulnerabilities everyone seems to get crazy and the first reaction is defensive, evasive or protective, like, it's just a different category of a bug, nothing else why is that everyone is scared of it? (We will discuss it later). For now, let's come back to our superhero and more Threat Modeling stuff. One nice thing that I like to say about Threat Modeling is that you need to think about the unimaginable.

The unimaginable threats

Sometimes you are going to face threats that you never imagined. Well, if you are not prepared for the easy ones, imagine the hard ones.

Antman destroys Iron-man's suit scene -Captain America Civil War 2016

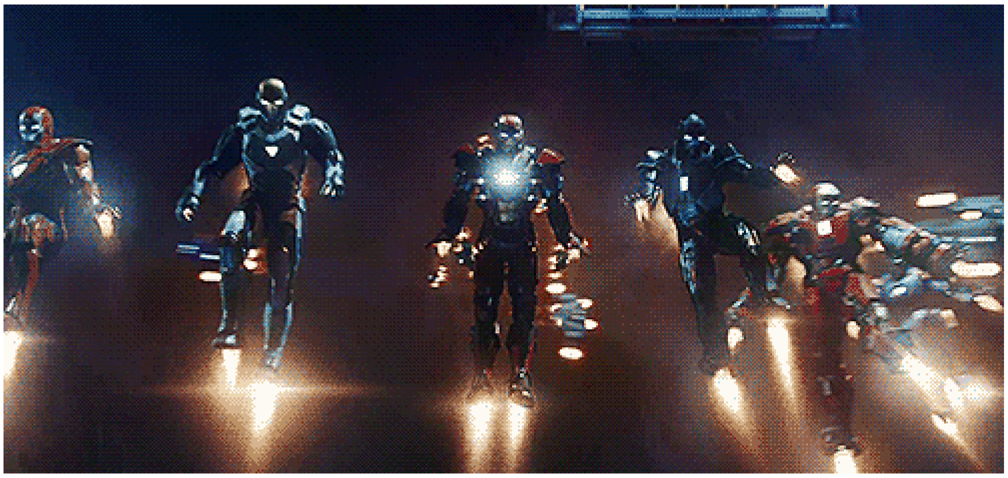
This is one of my favorite scenes, there is a lot to learn so stay focused now. First we can see that Tony's first reaction is to ask Friday what is happening when he noticed some failing in his hands thrusters, we will see in the future chapters of this book more details about Application Monitoring, but for now see that he does not ask anything for her directly, he just calls her name and she knows what to answer right in time. She then says: "We have some weapon systems offline", and Tony's answer is: "They what?", he is so surprised because he knows his "software", he knows his "product" because he designed it, so how could it have some "weapon systems offline" out of sudden? Then the funniest part when Ant-Man starts to talk to him: "It's your conscience, we haven't talked much these days." At this moment you can see Tony's desperate face calling for Friday one more time, and she decides automatically to deploy the fire suppression system. She also doesn't know for sure what is happening, they know the impact but what is causing that impact, no idea. Luckily the fire suppression system worked and expelled Ant-Man out of his suit.

One of the biggest responsibilities during a Threat Modeling session from the participants (specially the security ones) is to imagine the "crazy things", it's to think about the possibilities of attacks, even if they look impossible or funny sometimes, the thing here is, what is the impact? Can it destroy us or just tickle? Tony never imagined that a small man would first exist, second get inside his suit and "shut down" some systems, and it's ok because at the end of the day we can't think about all the possibilities, it's impossible by nature. But considering his work so far, he had a suit capable of handling it, if not he would be out of combat, considering also that Ant-man would not kill him, it was a fight of stopping each other not killing each other. And we finish with a few questions, how is your software today? Is it prepared for soft threats? How about hard threats? Can it handle a simple SQLi attack? Can it keep resilient during a DDoS attack? How about an insider that has access to the database? Think about that! Ah and by the way, he handled the Ant-man threat with the nanotech suit 🙂

When Threat Modeling does not work

A fix for each problem?

In Iron Man 3 we see the Iron Legion, basically a bunch of Independent Iron Man Suits, that were controlled by Jarvis and could act autonomously to support Tony Stark in any situation. He got to this point, because he was already obsessed with the fact that aliens now could invade earth as we saw in The Avengers, that's why he started improving even more his suits and power because he was sure that bigger threats were about to come.



It's clear that he tried to create a fix for each possible problem (threat) in Iron Man III with the Iron Legion, but it's impossible to scale at this point. Yes he was a billionaire, but still… how to build something for a threat that you don't know how it looks like? But still, a nice try that helped him a lot in the future versions, this was a very good MVP project for the future nanotech suit I'd say.

And here is the point, if you try to apply threat modeling to the whole system it won't work, you know why? Because the entire system is too much, and not everything in your system is critical, yes there might be some specific scenarios but in general the softwares around is critical in some specific functionalities. So, you only apply threat modeling to those scenarios where the significant security risk is visible and you need to protect, for example, you have an internet banking app and clearly the money transactions have more risk than just seeing the transactions history. Threat Modeling is a manual work, hard to automate and scale so, focus is very important one other advice I can give you is, go for the granular level like, take a functionality like login, or user registration to threat model, not the user journey workflow that is bigger and seems to have a lot of small functionalities within it. Also, do not threat model one function or method, it's nonsense.

Lessons learned, always

Good examples is what we need, we need to learn and have inspirations to be proactive, otherwise we will keep doing cyber security reactive and very often there is no time to react, for example:

1. In Captain America Civil War during the fight between Team Iron Man and Team Cap Vision hitted by mistake the Iron Patriot, he was aiming The Falcon but hitted Iron Patriot making him lose his suit source power which made him fall.

Tony tried to fly and save him from falling into the floor, but there were not enough resources at that moment, so the Iron Patriot a.k.a. Colonel Rhodes friend of Tony felt, and became paraplegic lucky because that fall could have killed him.

2. Later in Avengers Infinity War when the spaceship was coming back to space after invading earth Tony chased it and for a moment the spaceship was faster than him, he then asked Friday for a "support" and she gave him a boost, kind of a turbo flight speed mode, that made him able to catch the spaceship.

One more time Tony is showing us that the constant improvements are necessary, and they will NEVER be enough, there will always be a situation where problems, risk, damage will hit you and, how do you deal then? How much will it hurt? How much will it cost? He had a flaw (not enough flight power), he improved with the boost mode, easy right? I will ask you again, are your softwares are also like this? Every new version is improved from a security perspective?

Frameworks

There are few Threat Modeling frameworks out there, for all kinds of threat model you want to do you may find something that fits your needs, but here I want to bring you maybe, the most popular list of frameworks that you may want to look into:

- STRIDE
- PASTA
- LINDDUN
- Attack Trees
- Persona non grata (PnG)
- Security Cards
- Hybrid Threat Modeling Method (hTMM)
- Quantitative Threat Modeling Method
- Trike
- VAST Modeling

- Octave

Feel free to try them, at list a simple "Hello World" with each one will be helpful to make you understand their strengths and weaknesses, which ones might be easier than others, and which one might be a perfect fit for your process.

Conclusion

Threat Modeling is about imagining the unimaginable, the bad guys have no limits when they come after you, why should you have limitations regarding your Application Security process? Remember that after The Avengers, Tony starts to freak out with threats that the earth can suffer, because in Iron Man 1 and 2 the threats were "man", "technology", guns etc… but in The Avengers, fucking aliens came out of the space in a fucking warmhole in the middle of New York so, we need to be prepared right? Because of that came Utron (Avengers 2) and the Iron Legion (Iron Man 3), because he knew something worse would come at any time, and he must be prepared to save earth at any time from any threat.

I don't want you to freak out about security (maybe I do), but I do want you to really care about the software you deliver, how many people can be impacted by a security issue you might have? How the society may be impacted? Threat Modeling not only helps you create a culture of security, but directly affects the quality of your software in the end, since you care about security requirements from the beginning of your SDL.

Simple example of Threat Modeling

Imagine this Functional Requirement:

FR 1 - Login

The login page should be the first page that users see in the modified application. It should provide two text fields - one for entering a login name and one for entering a password. In addition, it should have a command button that initiates the password checking action. If either of the text fields is left blank it is an error that must be reported to the user. If both fields are filled in but there is no record of the username or the password is incorrect that must also be reported to the user.

Now, imagine some simple questions:

- Brute force attacks – There should be a captcha that must be solved prior to login.
- Encryption - Both username and password must be encrypted using the algorithm xyz, either during transport of the data or storage.

- Social engineering – The message of wrong login must not specify if the username or password are wrong to avoid telling that one or the other is correct.
- Log and monitoring – Each login attempt must be registered with at least minimum data considering the 5Ws, Where, What, Who, When and Why including the status (logins success or not).

The output of these simple questions are a few powerful security requirements that would make the software more protected from the beginning, if this login page is on production already then all these risks are also, if the security measures are implemented from the start, these risks are mitigated by default, that's the beauty of Threat Modeling, identifying problems before even the software exists. Even further, what if this "brainstorm Threat Modeling" happens based on the list of Functional Requirements? Then the Funcionation Requirement itself could be a merge of Funcional and Security Requirements, avoiding some "bypass" on the Security Requirements since they are "not business requirements" it's easy to "not do them" during the software project implementation.

Chapter 2

DevSecOps

"AppSec and DevSecOps are different things!"
– Cássio B. Pereira

Clearly Tony Stark realized that carrying a luggage, or coming to the "garage" to get his suit was not the best way to have it. It should be portable, even pocketable, easy to mount wherever he is, easy to carry. We saw in Iron Man III that the prototype Suit was able to have independent parts now, easy to mount, scalable, each part could literally go to him wherever he was and even he could choose to make the suit mount on somebody else. The same we saw in Captain America Civil War when he was wearing a simple watch that had defensive and attacking capabilities, and was very useful during the small fight against the Winter Soldier, he had a "flash bang" and a sound pulse to make the enemy stunned, plus the bullet proof material that saved him from being shot.

Let's think together here, what is best or easier, to secure a whole monolithic system, or different microservices? The point is, you **don't need** to protect your whole house, just the room where you store the most valuable assets. Or the whole house if it's the case, depending on the criticality of your business.

For Tony, the main asset to protect is himself, his life as a human being. Ok, the suit could be a chest protector for the heart and a helmet for the head? Yes, but still he would be vulnerable, so the whole body suit is more feasible. Lucky for our modern software development is not like this, we might want to protect the payments API (api.system.com/payment) than the list categories API (api.system.com/listCategories) for example right? They both are part of the same system, but clearly one has much more business impact than the other, PII must be involved and so on and so forth.

I will try to focus this chapter on the DevSecOps stuff, touching points of AppSec. The whole AppSec approach is regarding taking care of the Application itself, the software itself, the code, the .exe .jar files etc etc. The DevSecOps approach is protecting everything that is involved in the creation of the application, the whole supply chain (yes, not only 3rd party libraries) but the IDE, CI/CD tools, Ticketing System, Chats etc, everything that is part of the SDL. To understand better let's see the DevSecOps Manifest below and I also recommend you to check their website:

Leaning in *over Always Saying "No"*
Data & Security Science *over Fear, Uncertainty and Doubt*
Open Contribution & Collaboration *over Security-Only Requirements*
Consumable Security Services with APIs *over Mandated Security Controls & Paperwork*
Business Driven Security Scores *over Rubber Stamp Security*
Red & Blue Team Exploit Testing *over Relying on Scans & Theoretical Vulnerabilities*
24x7 Proactive Security Monitoring *over Reacting after being Informed of an Incident*
Shared Threat Intelligence *over Keeping Info to Ourselves*
Compliance Operations *over Clipboards & Checklists*

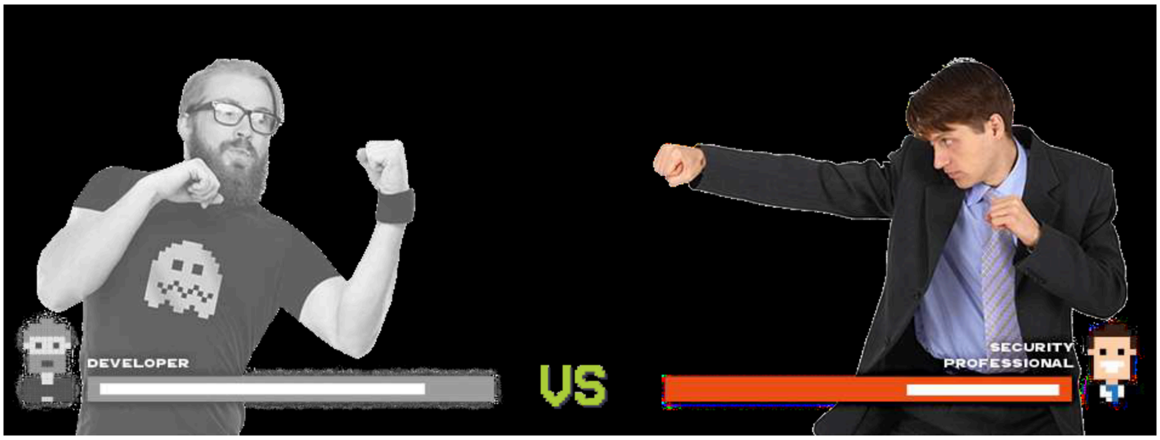
These nine points describe how things should be, they might be a little bit complicated to understand at a glance, but I'll try to go over the topics during this book. For now, I want to concentrate on ~~3~~ or 4 pillars that I consider for DevSecOps to really run smoothly:

- Culture (people) - To make / get everyone within the SDL to participate.
- Process (strategy) - To have a structured way of how-to.
- Tools - Otherwise, you won't scale.
- Automation - Automation is the key to make your AppSec / DevSecOps scale and work properly.

To understand a bit better, I created the following to make it clear how is the world nowadays if we do not consider the DevSecOps approach:

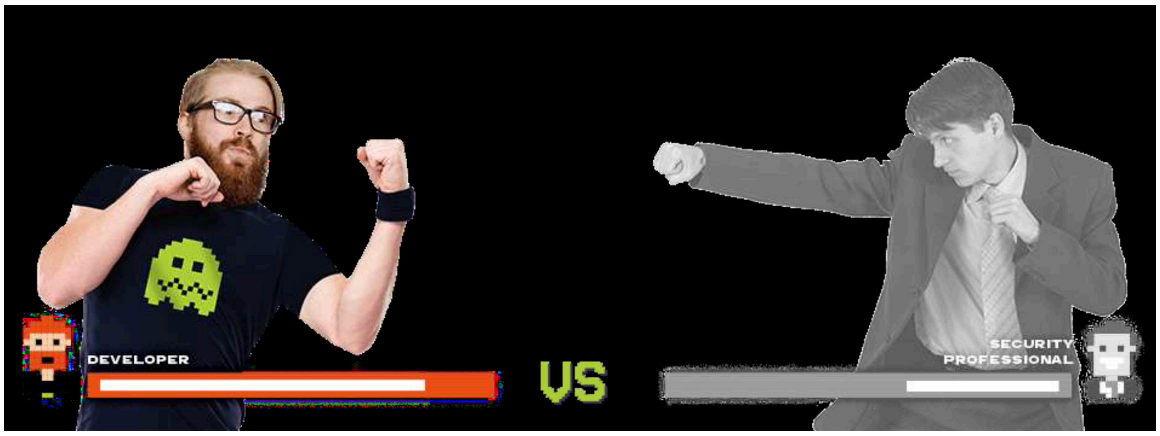
Before DevSecOps - Security Teams

- Say no to most things.
- Discover vulnerabilities in production.
- Doesn't know how to deal with other teams.



Before DevSecOps - Development Teams

- Does not prioritize the correction of vulnerabilities;
- Believes that the best solution to the problem is yours;
- Your code doesn't crash;



Clearly the teams does not talk or interact in a proper way, the whole communication and interaction is about power and ego, and developers and security guys can be fucking annoying with that, believe me I've been in both sides, so the DevSecOps philosophy is:

Culture - People 🤝

Good ✅

- Knowing how to say yes responsibly;
- Disseminate security concerns;

Bad ❌

- Wanting to participate in all decisions;
- Practice security by obscurity;

Process - Strategies ⚙️

Good ✅

- Start with simple processes;
- Always generate metrics;

Bad ❌

- Bypass processes for quick deployments;
- Wanting to automate everything;

Tools - Scale ⛰️

Good ✅

- Adopt Open Source solutions;
- See how other companies do it;

Bad ❌

- Not wanting to spend $$$ on tools;
- Wanting to use all available;

Automation 🤖

Good ✅

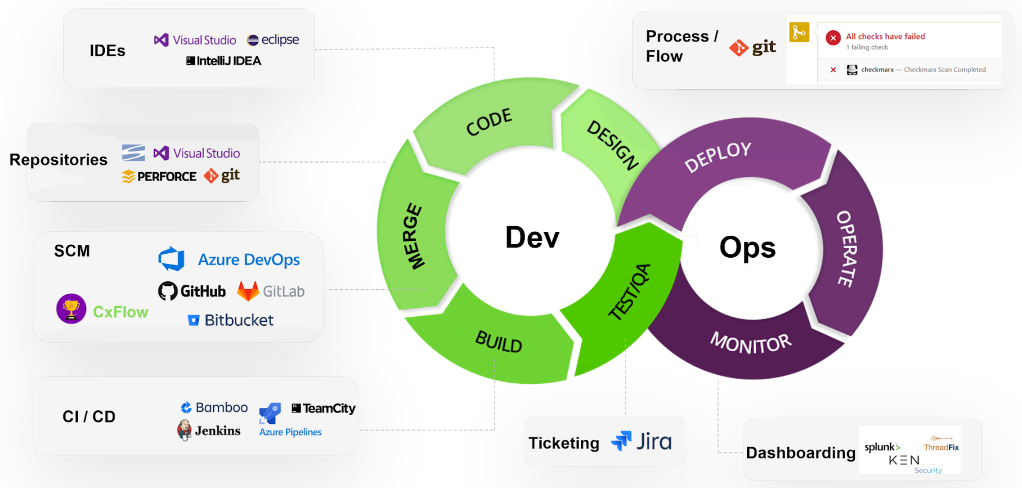
- Automating a whole flow, not part of it.

Bad ❌

- Trying to automate everything.

Automation as a critical factor of success

The whole SDL - Software Development Life Cycle is a complex ecosystem considering the number of tools, processes and integrations in place, the scale of it is already unmanageable, on the following picture you can see in a high level parts of the SDL:



And without some automation, the simple goal of creating a simple software can become a nightmare. From the IDE to the Cloud, through the ticketing system, SCM, CI/CD tools, dashboards and flows, how can everything work manually? I'd say it's a mix, automate whenever you can but you will survive with one or two manual tasks.

Conclusion

DevSecOps is part of AppSec, which means DSO is another area that requires attention. Is not only security tools integrated in the pipeline, or automatic scanning your PRs, is much more for example:

- These tools properly configured? Your AST scanners?
- The SAST has a proper rate of false positives? Did you check it? How about the true positives?
- Is the SCA resolving the dependencies properly? Not missing anything?
- Do the pipeline steps have proper access management? Who can disable a security step?
- Do you collect KPIs of your AST scanners?
- How is the developers' machines hardening?

Chapter 3

Vulnerability Management

"Know your problems!"
– Cássio B. Pereira

I'd dare to say that Vulnerability Management is where most companies fail implementing an AppSec program. This is just a regular process and, still, companies neglect it. Why is it so difficult tracking a bug / vulnerability? Let's imagine a simple flow, the security team or whoever in the world found a vulnerability and reported it to the company, someone validates it and addresses it to the team / person responsible for fixing it. There must be some prioritization, it can be something very urgent or something that can wait X time. Then, a fix is applied, someone tests it, deploy to prod, done. Simple right?

I think that at this point it's pretty clear that Tony Stark handles his shit. Not by mistake every new suit, or Mark, has an improvement. Either to the defense power or attack power, sometimes even both. Let's remember when he fought Captain America and the Winter Soldier in the Civil War movie, his targeting system was damaged and he was not able to shoot properly, forcing him to do the "manually" aiming. When later Iron Man fought Thanos in the Avengers Infinity War, he was now able to fire remote guided missiles, without the need of aiming even.

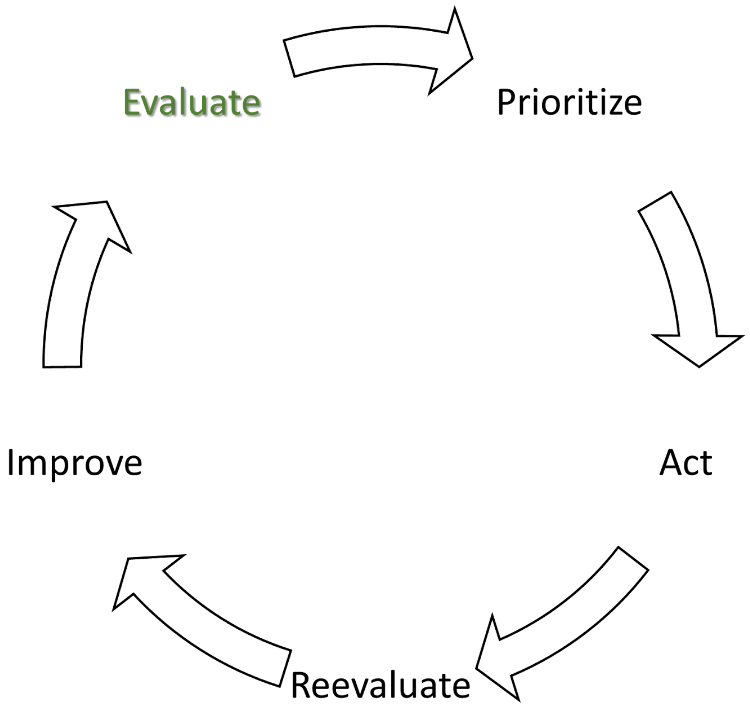
How about your software? Do you have a bug list? How about a vulnerability list? And a security improvement list? Security requirements maybe? Ok no worries…

Vulnerability Management Process

In order to achieve the goal of really managing your vulnerabilities, you need to prepare for that, you can't monitor / handle all assets in your company, most of them are not even critical enough to make you spend time / money on them (for this purpose). So, first you do a Preparation, that is composed by:

- Determine scope;
    - What is going to be part of your vulnerability management.
- Define roles and responsibilities;
    - Who will be responsible for what?
- Select assessment tools;
    - What are going to be your scanners?
- Define policies;
    - What are the rules for the results?
- Identify assets;
    - Which systems, repos, etc are you going to scan?

Then… you apply this simple flow:



I will explain one by one using a SAST scan example:

- **Evaluate (scan)**
    - Run the SAST scan in your repo.
- **Prioritize**
    - Get the results and prioritize them, for example, the results of the SAST scan was only 1 HIGH vulnerability. But, it's an impact on a legacy system that has no critical business impact for you, so you can tag it as MEDIUM.

- **Act (fix the problem)**
    - Address this vulnerability using your ticketing system for example, and set the expected SLA considering the MEDIUM vulnerability.
    - Developer will fix and make it available for testing.
- **Reevaluate (scan again)**
    - Scan again the same repo, with the same configurations.
    - The vulnerability should not be there anymore.
- **Improve**
    - Think about ways of automating:
        - The scan
        - The results track
        - The results notification
        - The fixing time
        - The re-scan
        - Etc
- Repeat…

This macro process works pretty well for any security scan / testing with slight changes according to each test if really necessary, but the whole idea is the same, you test, prioritize, fix, validate and improve.

Tools

You can use any bug management system, task system, ticketing system etc that will work the same, the secret here is to centralize your vulnerabilities in one place, you might have a SAST, SCA and a DAST scan that you want to consolidate the vulnerabilities to optimize the management, you can't have 3, 4 ,10 different places otherwise it won't scale and it won't flow properly. One nice tool to use is Defect Dojo from OWASP. Or you can still use:

- Jira (Popular)
- Excel (😶‍🌫️)
- Github Issues (Cool one)
- Notepad! (Just kidding, **or not**)

Reference

I'm not Tony Stark (yet) so I did not create any of this, I just searched and applied what is, in my opinion, the simplest process to do Vulnerability Management, here is the reference and here is the PDF.

Chapter 4

Application Monitoring

"Know your application before your enemy."
– Cássio B. Pereira

Sir, Take a deep Breath Iron Man 3 (Clip) 2013

Note that in this scene Tony did not ask for Jarvis help, Jarvis KNEW WHAT THE FUCK TO DO IN THE RIGHT MOMENT, you might wonder why am I screaming? BECAUSE FUCKING DEVELOPERS, DOES NOT EVEN KNOW THEIR OWN SYSTEMS, and this pisses me off. So I will ask you a few questions:

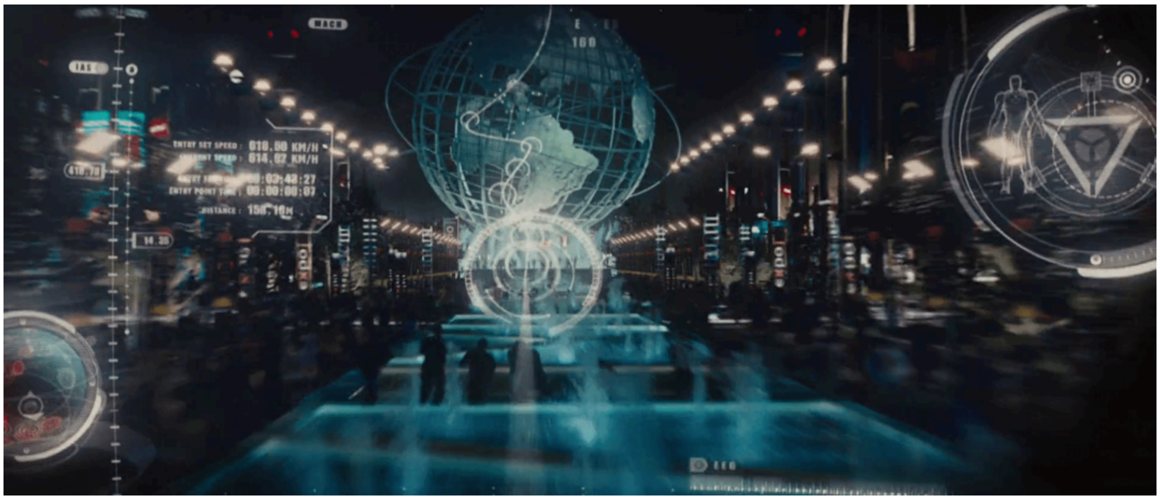
- How does your application behave under attack?
- It just fall?
- Do you have any logs?
- Automatic alerts?
- Do you even know that you are under attack?

After two decades of working in the industry I had the chance to meet a variety of people and different businesses and without any regret I can affirm that 90% of developers do not even know what they are building, they know the task, the feature, but the whole? No. They know the product, the name, the goal yes… but when you ask about the application as a whole, at least a macro view they have no clue. And, that's ok, they must focus on their "task, feature, ticket etc" not the whole, but, during a security incident when the DFIR team must act to either contain the threat or just to investigate, nobody has a fucking idea of the whole system, what are the services, witch on talks to each one, where is the database hosted, who is the administrator, who is the product owner responsible for decisions… So many questions that in theory are easy to answer, but during a crisis, DFIR teams are ALONE, they must do an internal FORENSICS first to know all this, then they can start digging into the incident itself, and this much time usually is very critical.

So my point here is that collaboration from developers should be much more taken seriously, they know how the application works / was built, they must know, not only to make their own work easier, but also to support the whole company structure regarding security posture. I once had an incident at a company that I worked for, some APIs were being abused by bots, raising the requests rate to 500.000 in an hour during the morning. The First thing was, give me the logs, I want to see the request IP, geolocation to understand the scenario, there were no logs. Ok. Turn on the logs, WAIT FOR THE BOT TO RUN AGAIN, basically, wait to be attacked again so i could have data to check on, could not only block requests, block what? Then the BOT again, IP addresses coming from Amazon… After two weeks of trying to understand (and I had the suspicions already), during an informal coffee with a developer, he mentioned that his TESTS ON APIs were fine, I asked for details and for my surprise (or not), was him just testing the APIs there were "under attack". But until then, no one knew anything about those APIs, who had the URLs, who had access to it (they were public, but suppose to be for some customers only), who was in charge of development (few teams were), so many simple questions, that in a company without the minimum of organization and discipline, during a crisis, can be fatal.

Do you know what is the normal behavior of your applications?

Tony Stark knew (with the support of his A.I. assistant of course) at any time, a bunch of useful information, he had all the kind of data regarding his suit of course plus a lot of data about the external environment such as air control traffic, climate, earth perimeter etc. Check this next scene, in real time Jarvis defines a route of flight to outrun the drones, even making some of them crash:



It is a perfect example of application monitoring, you must have the information in order to know what is happening and decide what to do, especially if you are under attack. And not only having the information, but the right information at the right time.

Information is the key

Information is EVERYTHING for the correct decision making. Tony's suit is connected to several data sources, JARVIS knows everything to keep him informed at any time, and sometimes without asking for it, just a few things Jarvis "HAS DOUBT". Not only trusting data sources, but also, processing real time information like in this scene:

Iron Man Plane Rescue Scene - Iron Man 3 (2013) Movie CLIP HD

See that during all the time Jarvis keeps him informed of the altitude and is showing him on the screen the "route plan" to grab each person? This is so fucking amazing and a good example of the correct data at the right time during a crisis. And all that, with a remote controlled suit, Tony was not wearing it as we can see at the end of the scene. See how fast sometimes we need information in order to make a hard decision?

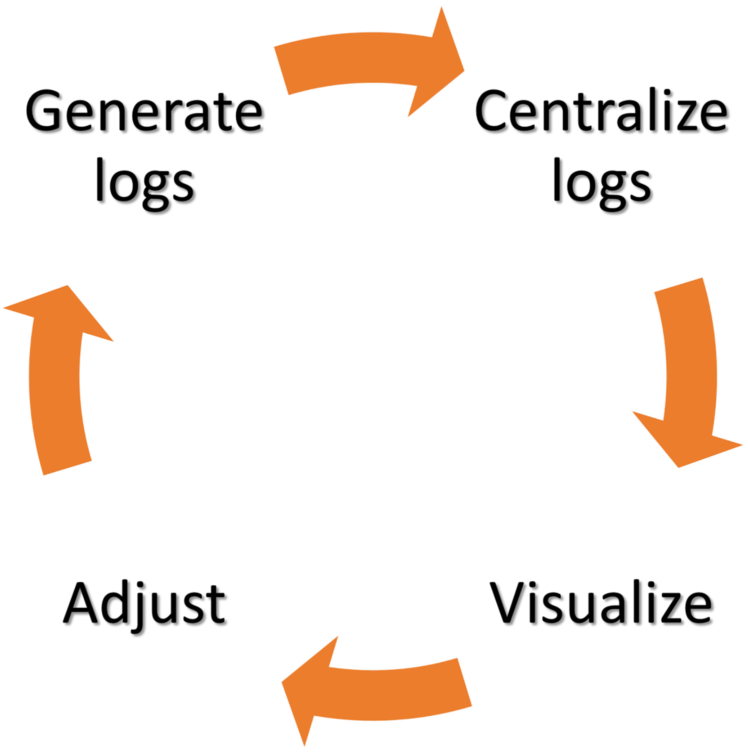
Logs

In computing, data log is an expression used to describe the process of recording **relevant** events in a computer system. This record can be used to **restore** a system to its **original** state or to let someone know its **behavior** in the **past**.

It's all about **behavior,** knowing the behavior of your software is essential for adequate protection, only with continuous monitoring is it possible to know the behavior of your software. Once you know / determine the normal behavior, it's easy to know / determine the anomalous behavior, for example:

Your authentication API has 500 requests daily based on data observed during 30 days. We can assume that this is the normal behavior of your authentication API, the amount of requests daily is 500. Then one day you have 500 per minute on your API during the first hours of the day, what is the first thought? What is the first reaction? Clearly something **not normal** is happening, you will dedicate some time to understand and react / take a decision about what to do, but without this data, it would be just a "normal day", maybe the system would be a bit slow but, nothing much to worry about.

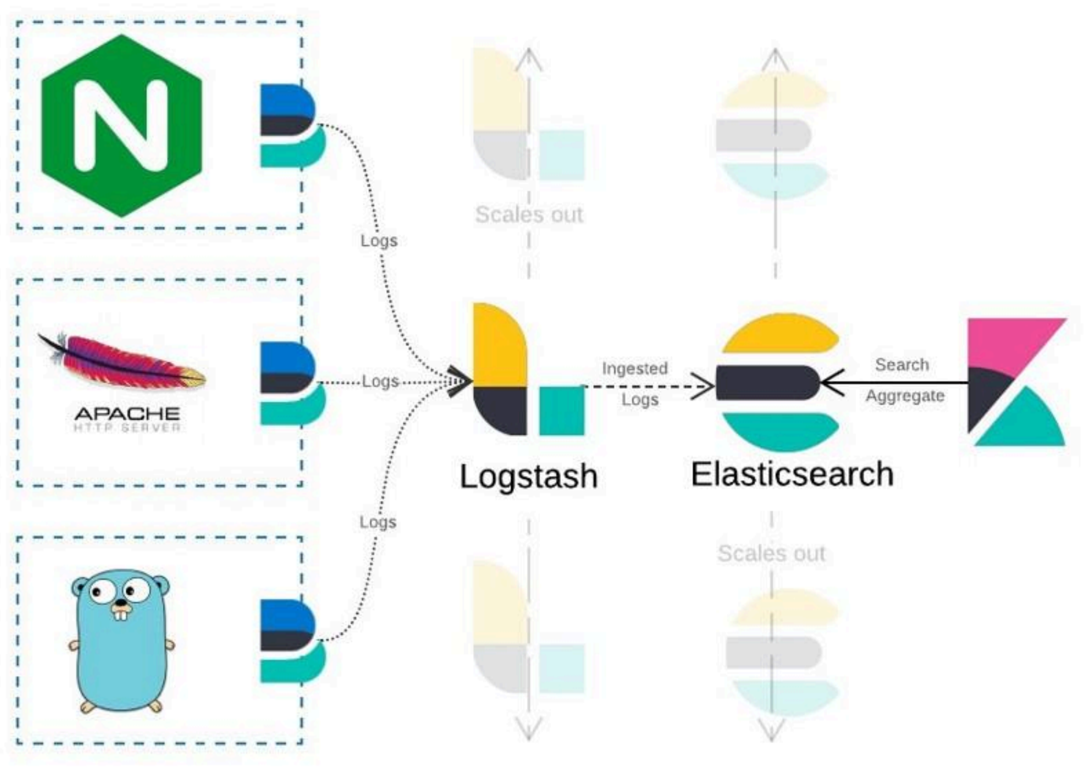
Managing logs is simple, even do it might be expensive:



- **Generate logs**
    - Make your application generate the logs you need, also the perimeter architecture like: firewalls, wafs, cdns etc.
- **Centralize logs**
    - Store these logs in a central way, easily accessible and easy to manage and correlate.
- **Visualize**

    - Create views and dashboards that present the log data in a friendly way, reading logs might be time consuming and not pleasant.
- **Adjust**
    - Delete irrelevant data, add relevant data. You might want to know the IP address of a request, but not the size of bytes depending on the context.

A very easy way to do this is using the ELK Stack, but you can do this using any other logging or SIEM solution:



Chapter 5

Application Security Program

"Security is everyone's job."
– Cássio B. Pereira

Points to consider

We all know by now that Tony Stark not only got "crazy" about his safety and protection, but also about the safety and protection of the planet earth, especially after The Avengers when aliens came out of a worm hole in the middle of New York city. He then, started to contaminate everyone with his ideas, Ulton was the project of creating a "Suit around the world", that failed yes, but still a try 🙂, so to avoid your AppSec program failing, here are some points to consider:

- Define critical projects;
    - What are the real critical projects to your business?
- Communicate the teams;
    - They must know about the AppSec Program, how, when and why are the minimum information that you need to provide.
- Conduct training;
    - Each audience has specific needs, developers must learn how to fix a vulnerability, but Q&As must know how to exploit them for example.
- Appoint the Security Owner;
    - It must have an Owner, someone within the team, squad, tribe whatever that is responsible for the Security Hat.
- Define baselines;
    - What are the minimum requirements that you want to implement? For example, all code must have a SAST scan prior to merge to the main branch.
- Define acceptance levels;
    - What are the risks that you accept? For example, you can define that all HIGH and CRITICAL vulnerabilities must be fixed prior to deployment.
- Define CI/CD strategy;
    - Security must be part of it, doesn't matter how you do it, include security on your CI/CD. Scans, checks, tests are examples of tasks that can be part of it.
- Generate metrics & KPIs;
    - It's important to know how your AppSec Strategy is performing, defining and collecting KPIs is important to measure that.
- Gamification
    - Sometimes it is helpful, even if you consider that developers should not be rewarded for fixing vulnerabilities for example, because they should not create them in the first place, it's a tool to consider for sure.

Coming back to Tony Stark he thinks security, he breathes security, he lives security, he is neurotic about security. Usually this is the mindset of the Security team, and it’s wrong. **Everyone is responsible for security**. Imagine the following scenario, you live together with someone, one of you leave the house earlier then the other, but, someone left the front door open, someone went inside during your absence and stole all your assets such as TV, Video Game, Sound Bar, Laptop etc a huge financial loss for both of you. Whose fault it is? Actually, who left the front door open? You who left first? Your partner that left after? If you, why did not your partner check then? Do you realize that it doesn't matter? In the end the loss impacted both of you, all your valuable assets were stolen, if you or your partner left the door open is the minimum of your problems.

I hope that by reading this book you will find out that you are also responsible for security, given your role and position more or less responsibilities or tasks that's for sure, but one way or another you are not excluded from the security team, you might leave the front door open, you might not check if someone left the front door open, but you have responsibility too, and the next scene explains why:

Iron Man 3 2013 ► Tony Stark vs Ellen Brandt Bar Bud Light Scene ► Movie CL…

Tony improvises when needed, even without his suit he was able to dismiss a real and hard threat. His life was in danger, so one more time, with the tools and conditions that he had, he handled the situation. I often see SDL Engineers (anyone involved in the SDL) but I don't have IDE X, I don't have the budget Y, I prefer to use the framework Z blah blah blah, all excuses to not make the right thing, all "ways out" of assuming the security responsibility inherent to their role, let me tell you something, fuck your IDE, framework, budget, there is always a way of doing it, so, do it.

Chapter 6

Why?

"Crackers, cyber criminals, cyber war…We are just an ordinary company."
– An ordinary company

Once Tony Stark said:

"Gods, aliens, other dimensions…I'm just a man in a can."

Stark, Tony - Iron Man 3

Once few companies said:

"Crackers, cyber criminals, cyber war…We are just an ordinary company."

An ordinary company

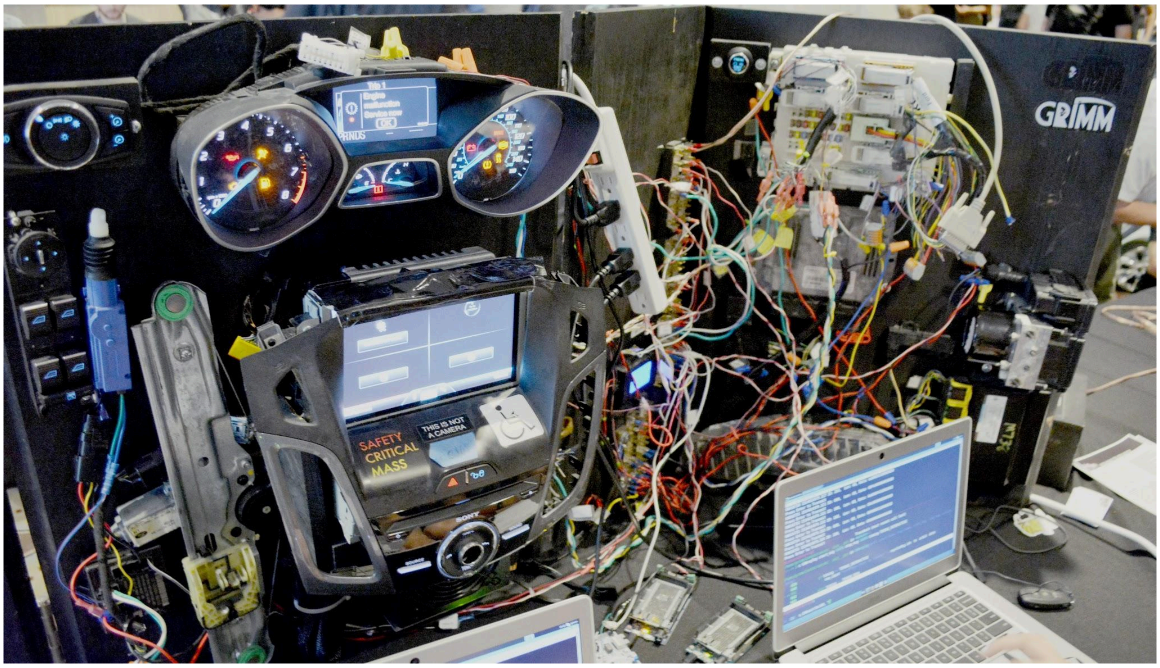
And this mindset has taken over the minds of millions of executives, engineers and professionals around the software (I.T.) industry, witch is kind of ok if you analyze the nature of some business, like a simple static website to show what a company does might not be a target for a cyber criminal at a first glance, compared to a banking application where billions of real money transactions are processed daily, but from the perspective of an organized cyber crime group, both are equal and has, maybe, the same earning potential.

And you might think the same, Cássio I do work in a software company but our software is not critical, it's just a warehouse management system for marketing materials like pens, banners and swags, what can go wrong? It's not about the company operation in this case, but about the data that this company owns, if I want to harm someone, I can be interested in the events that this company promotes, since you store the marketing material, when a big request is done something might happen, so I can track to understand the operation in other to achieve what I want as a "malicious actor". Or I can just focus on diverting materials to sell for example. Always remember that, for malicious actors, there are no limits on what they can do.

So, your company might be a target just because it's a brand attached to some marketing that went wrong, or a target by the assets itself such as money, materials or valuable resources, sometimes for positioning publicly to the government side A or B or even due to a public case where this company might did something bad to a former employee. It doesn't matter why (in this case), but you are a target for someone, you might not just be found, yet.

Why? Well…

Some people might not realize yet, but our fucking life is controlled and guided by software EVERYWHERE, from the chain of production of food, to the payments and shopping we do, there is a software somewhere, so protecting software nowadays is not anymore protecting yourself pockets or businesses, is protecting the reality and the society itself.



If your car has a software failure, so what? No need to be the news models no, any old car with Electronic Fuel Injection is already done by a software. So if you are a powerful executive in a big company, or a member of the government, you might be a target for someone that might want to harm either you or your business somehow.

Conclusion

Maybe your company is not a critical infrastructure provider or a business itself, but what impact would it bring to the society (peoples life) if you have an incident right now? Think about it next time you code!

- Some people lose money?
- Some people won't be able to pay for something?
- Do your company employees lose their jobs?
- What else?

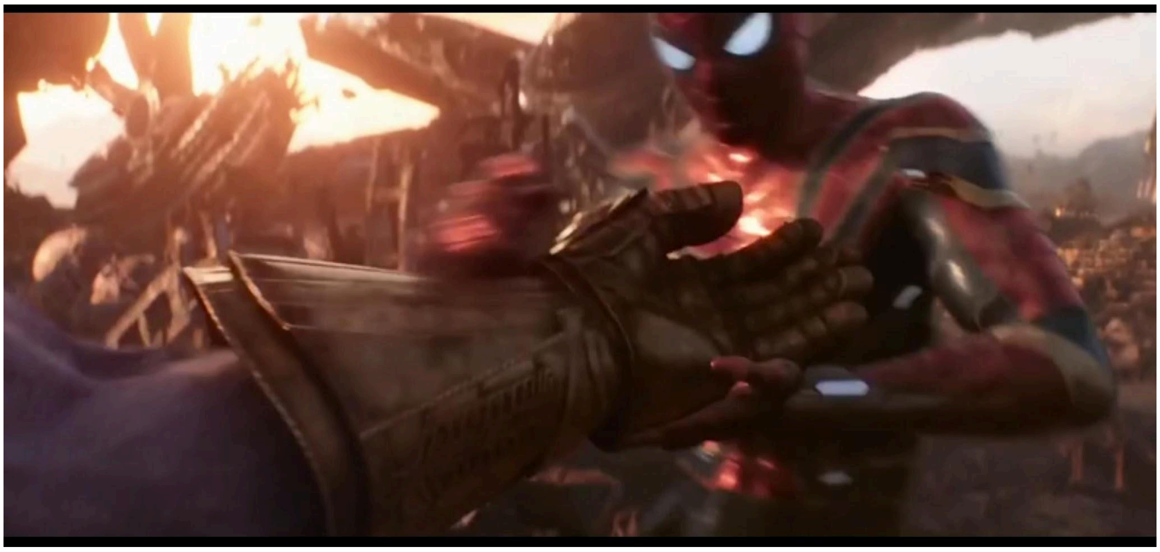
Remember:

- Everything is controlled by software;
- We need to build a better society;
- We need more individual protection;
- Because our lives depends on it.

Chapter 7

Bônus

"Because I really like Iron Man."
– Cassio Pereira



In this scene in Avengers Infinity War, the team tried to take Thanos' Gauntlet and almost succeeded, but in the last moment he woke up and got it back.

Knowing this, Tony Stark prepared his nanotech suit to be able to incorporate to and from any other material, so when he had the opportunity to take Thanos' Gauntlet again on the Avengers Endgame, he managed to do it just by touching it.

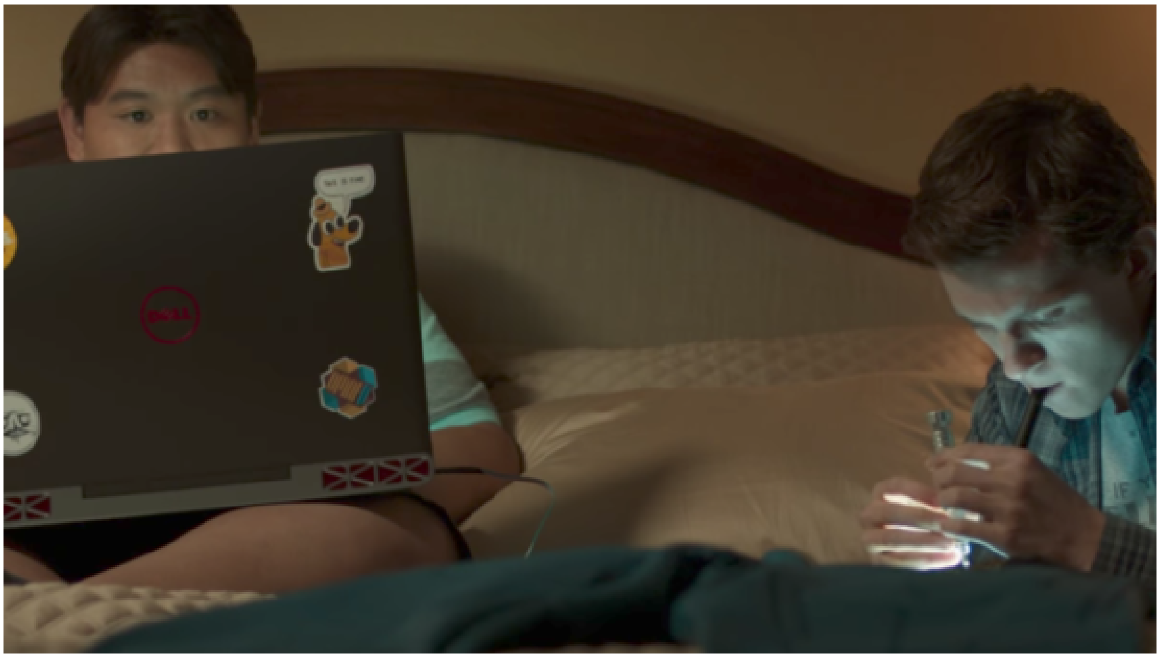
We see more of this capability after on Spider-Man No Way Home, when Peter (actual one) got attacked by Dr. Octopus and his tentacles touched Peter's nanotech suit, the nanotech automatically incorporated into the tentacle, making it able to be controlled by Peter.

In Iron Man I he was kidnaped by the terrorists, kept in a cave where it was impossible to track and find him.

Then, in Iron Man III his suit was now able to come from anywhere, so there was a tracking system, plus the capability of being armed with his suit, or at least part of it as we saw in the movie. Also, he was able to improvise, since his suit came part by part, he had to fight to make himself free even without the full suit.

And let's not forget the powerful watch that he had, that moment the suit was not allowed to wear, since he submitted himself to the Treaty of Sokovia that originated the Civil War. But still, without his suit, he was not fully vulnerable and unarmed.

Spider Man also had a tracking system in his suit, when Tony Stark invited him to the Avengers and made him a new suit, he wanted to protect the kid (since he knew what could happen). Of course, Peter hacked the suit and removed the tracker to do some shit, but ok.



On Iron Man I the Obadiah used some kind of Sound weapon to make Tony paralyzed, you can see it here Obadiah Stane Steals Tony Stark's Arc Reactor Scene | Iron Man (2008) Movie CLI…

Later in the Avengers Age of Ultron, "same sound" affected everyone in the room but not Tony, you can see his reaction in the first 5 seconds of the following video, not needed to watch all: Avengers vs Ultron First Fight in 4K Ultra HD [Fight Scene]

Chapter 8

The End

"Pay attention to this scene!"
– Cassio Pereira

Tony Stark & Captain America Argument Scene - Avengers: Endgame Movie (Scene) | M…

Here is the transcript of the above scene, and I will comment later, but watch it again! The most important parts are in **bold:**

- 00:00:00.080 It's been 23 days since thanos came to
- 00:00:02.080 earth
- 00:00:04.720 world governments are
- 00:00:06.319 in pieces um the parts that are still
- 00:00:08.639 working
- 00:00:09.679 they are trying to take a census and it looks
- 00:00:11.840 like he
- 00:00:14.920 he did exactly what he said he was
- 00:00:17.199 going to do thanos wiped out
- 00:00:21.359 50%
- 00:00:22.720 of all living creatures
- 00:00:28.560 Where is he now? where?
- 00:00:30.800 we don't know
- 00:00:32.640 he just opened the portal and walked
- 00:00:34.640 through
- 00:00:38.160 what's wrong with him
- 00:00:40.160 uh he's pissed
- 00:00:42.640 he thinks he failed
- 00:00:45.200 which of course he did but you know
- 00:00:46.719 there's a lot of that going around ain't
- 00:00:48.079 there honestly until this exact second I
- 00:00:50.559 thought you were build-a-bear maybe i am
- 00:00:52.320 we've been hunting thanos for three
- 00:00:53.840 weeks now deep space scans and
- 00:00:56.960 satellites and we got nothing
- 00:01:00.800 Tony you fought him who told you that?
- 00:01:03.280 There was no fight no he went my face with
- 00:01:05.680 a planet while the Bleecker street
- 00:01:07.200 magician gave away the store that's what
- 00:01:09.200 happened there's no fights okay he said
- 00:01:11.360 did he give you any clues any
- 00:01:13.040 coordinates anything uh
- **00:01:17.040 I saw this coming a few years back I had a**
- **00:01:19.119 vision I didn't want to believe it**
- **00:01:21.040 maybe I was dreaming** Tony I'm gonna
- 00:01:22.880 need you to focus **and I needed you**

- **00:01:25.759 as in past tense that trumps what you**
- **00:01:28.720 need it's too late buddy**
- **00:01:30.560 sorry**
- **00:01:32.240 you know what I need**
- **00:01:33.520 i didn't shave**
- **00:01:35.840 and I believe i remember telling oh yes**
- 00:01:38.400 Tony Tony Tony… **alive or otherwise that**
- **00:01:41.119 what we needed**
- **00:01:42.880 was a suit armor around the world**
- **00:01:44.880 remember that? whether it impacted our**
- **00:01:46.720 precious freedoms or not that's what we**
- **00:01:49.520 needed** well that didn't work out did it
- **00:01:51.280 I said we'd lose**
- **00:01:52.880 you said we'll do that together too**
- **00:01:55.600 and guess what cap**
- **00:01:57.040 we lost**
- **00:01:58.719 and you weren't there**
- **00:02:01.200 but that's what we do right our best**
- **00:02:03.040 work after the facts what are the**
- **00:02:04.560 avengers we are the avengers not the**
- **00:02:06.960 pre-avengers** okay right you made your
- 00:02:08.878 point just sit down okay okay no no
- 00:02:10.479 here's my but you don't want me to just
- 00:02:12.000 sit down we need you your new blood
- 00:02:14.560 bunch of tired old meals i got nothing
- 00:02:16.959 for you cap i got no coordinates no
- 00:02:19.599 clues no strategies no options zero zip
- 00:02:22.800 not a
- 00:02:24.080 no trust fire
- 00:02:30.160 here take this
- 00:02:31.680 you find him you put that on
- 00:02:34.160 you hide
- 00:02:37.040 I'm fine
- 00:02:38.879 let me…

Now let's discuss this iconic moment of the MCU, where Tony is basically saying: I TOLD YOU MOTHERFUCHERS! And it reflects 99.99% of the Cyber Security teams nowadays, we warn about some threat, risk etc and nobody cares, then we shit happens they come to us screaming in panic, and we have to fix…. but I wish I could just say, I TOLD YOU.

Unfortunately that's the reality, the job of a Cyber Security team in general is to warn about the threats, that might be a risk at some point, that might exploit a vulnerability and become an incident, yes it's possibilities that might never happen, and to be more precise on this possibility a lot of tools bring nice capabilities, together with a nice business impact analysis, you can basically focus your efforts on fixing what matters, instead of trying to deal with hundreds or thousands of vulnerabilities around.

In this scene, Tony is making a reference to Ultron that initially was an idea of "suit around the world" to protect it, as we know didn't work properly and there was a mass, but after kind of worked, with the creation of Vision on the same movie. But we all know that Tony started saying this right after The Avengers and the invasion of New York. Due to the fight because of Ultron, that Captain America didn't want it together with others that's what Tony is mentioning now, because now Thanos came and destroyed everything, and Tony clearly knows that with more time and tests, Ultron would be the perfect weapon / shield against Thanos, the biggest threat so far. We also see in the What If series, that Ultron kills Thanos in 1 second!!

Another very important point in this Tony moment is that he mentions "we needed… Was a suit armor around the world remember that? **Whether it impacted our precious freedoms or not, that's what we needed!**" This is a moment for reflection, because once Ultron idea came out, the worries of others was exactly the "freedom", that Ultron would be monitoring everything all the time, kind of.

Here the parallel with AppSec is very clear, nowadays developers (and others) have all the power to run a company and its decisions somehow. If they don't want to run code scans, the company easily accept that, if their CI/CD process is slow due to a security testing, remove it, if the IDE plugin that checks the code quality is no "fancy enough" they scream and remove it, if the DAST scan on a Q&A environment is not performing well, why we need it? If a pentest is needed to run, they ask why? We are not targets. Sometimes when we are able to run a SCA or SAST scan, and bring them some vulnerabilities to fix, their first reaction is: it's a false positive or worse, it was always like this, why do we need to fix it now? The thing is, everything is priority but security, and at some point I agree with that, risk taking is part of human life since the creation, but that doesn't mean being negligent, doesn't mean that what only matters is money and the business running (yes it is, but not only).

And what I most like about Tony's speech is when he says: "**but that's what we do right our best work after the facts,** we are the **avengers** we are the **avengers** not the **pre-avengers**", this should be the Cyber Sec slogan, *we act after the facts,* more specifically the AppSec slogan, since the whole job (or majority of it) is to prevent and protect, we end up always cleaning shit after something happened. And more funny, when something happen, the whole company call the Avengers (Sec team), and sometimes works, sometimes the incident is not so critical that the Sec team can handle, but very often is not, very often the company is on the world news because of a security incident that impacted not only the company, but the society itself. While I was writing this book, there was the CloudStrike + microsoft incident, not even a security incident just the old and good bug, that caused a outage on the CloudStrike agent for Windows hosts, the whole fucking world stopped due to this bug. Just as an example of society impact, flights stopped / got delayed, payment systems, stores etc anything running on Windows using CloudStrike agent, stopped 🙂

Look, I was a software developer for over a decade. I created code for Financial companies, for Tourism, for Retail, for e-commerce, for Marketing agencies, for Logistics and a few more. I know how the SDL process and how companies really focus on delivering and keeping up and running, yes ok. But if a fucking security issue, only one got exploited, the business might not be up and running dammit. So, since I migrated my career to AppSec, I've been trying to make companies and individuals realize this importance, not only scanning and finding / fixing problems, but also stopping creating them in the first place. To run an AppSec program I'd say, it's easy (it's not), to buy a SAST / SCA and start scanning? 5 minutes job, to fix? Maybe an hour, to stop creating this code problems? Maybe the whole life journey of training for a developer, to be able to, at least, think before setting up a parameter without validation.