1p

Hello today we're going to talk about software security risk management in the context of our project,

2p

So the learning goals for today are to review different security focus development processes with an eye towards risk management how they deal with security. We've already done some of this in the preliminary lecture if this would be much more focused on security and again keeping with the theme of the class, we will make the argument that **security must be handled and addressed at every step of the software development lifecycle.**

3p

So recall that adding security to a completed system just throwing something behind a firewall or trying to retrofit assistant to be more secure is usually a very superficial way to deal with security, **security is really a systemic property** that permeates all phases of software development, you have to understand what specific threats, you are facing your system, you have to design controls and strategies to deal with those threats. And you need to consider things like recovery and how you will continue operation or reestablish operation after a security incident.

4p

Security management processes are really just things that we do to advanced projects, so this involves things like sequencing activities and scheduling activities, developing a system in a predictable way, so, while it's a basic truism that will never completely mediate security concerns in the system. What we can do is attempt to identify and respond to the most significant threats we believe will be facing. processes are created equal they're also not written in stone organizations tend to develop strategies or adjust strategies to fit their specific needs.

5p

So one thing that we should do early on, is think about the goals of the system. So in general quality goals are often framed in general terms. and I wouldn't get away from this, you might say, like things, like satisfy, the customer achieve all milestones incremental requirements, you might have more even have some metrics to finding your project that you are working towards close all of a defect, make sure all tests pass. **But remember that these generic milestones don't mean a whole lot on their own,** **I think it's a better idea to have more specific quality goals like mitigate threats identified in the threat model, pass some measure of security testing or penetration testing, something that you can actually tie back to in your project.**

6p

**So a quality process is really inseparable from a development process.** I talked about quality processes what I really mean is that we're able to develop processes to evaluate the qualities present in our system of which the security is one. It's also inseparable. We talk a lot about in software engineering. We talk a lot about having software development strategies. Software development lifecycle, software development processes, but really, fundamentally, **people do this work.** So I think it's very important and understate an underrated aspect of any successful organization to have good training and good shared understanding of what quality and security mean in a project.

7p

Regardless of the process you're always going to have some core properties. Processes are geared towards addressing common issues. So as we look at software development strategies to focus on security, they're going to see some common approaches like dealing with specific threats are doing risk assessments we're doing. We should be mindful that we're not building software systems in a vacuum. Here we're working on projects in the context of a business. **So security is rarely the end goal for a project. Security is a supporting goal**. It's one of those things that you need to have but doesn't in case something goes wrong. But the success of the project is not often not predicated on having a secure implementation. Quality processes security evaluation processes need to be **timely**. So you were able to find things as close to when they were introduced as possible. You need to have some **visibility** into your processes. And any like anything if you can't **measure** it. It doesn't really exist.

8p

So we've been over a number of different SDLC. Software development life cycles. And we've seen you should have seen in the remote lecture is a various different strategies. Some are sequential, some are we've looked at the V model, some take a V model approach that high quality control activities to sequences in the waterfall are in the process, some are iterative somewhere incremental some offer very lightweight advice, some are very prescriptive really takes all kinds of really has all the number of different flavors.

9p

Choosing a particular process will depend on things like team size, product complexity. There's a paper by a guy named Justin Rockwoord. He's at Carnegie Mellon graduate, who did a technical report on choosing software development lifecycle based on the number of criteria.

10p

And the paper, I think, is worthwhile for any software engineer to read. But it doesn't really address. But to address security, we have a number of software development life cycles that will work well for certain types of projects. But what about security specific processes. How do we deal with security in a more explicit way during software development what will be important for you in the first part of the project for our course.

11p

Well, perhaps the most famous security focused development lifecycle out there is Microsoft's SDL. I know we've been over this to an extent. It is more or less reflects the waterfall approach and is a real less a tightly connected process. But I mean by that it's less of a roadmap and more of a collection of activities that they need to be conducted. During different phases of development, so what I like about it is that even if you don't select in implement the SDL completely, you can look at the practices within each area, regardless of other what process you're using and you can say analyzing the attack surface going into a design that can be a good idea. For example, conducting fuzz testing during verification, is a good idea. And then I really like the idea that at the end of the SDL. There is an explicit phase where you develop an incident response plan for how to handle the inevitable scenario of a flaw defect or security compromise occurring.

12p

Agile processes are have been invoked for the last decade or so perhaps a little bit longer. All these really are a collection of software development. Strategies that do away with a lot of ceremonies. So when I say ceremony, I mean rigid prescriptive advice. So waterfall which is the canonical example of a rigid process has you do things in a particular order. You do requirements once requirements have stabilized and are that ID, then you move on to design and wanting to design is complete and verified you move on to implementation, and so on this works well for some types of software development projects. Notably when it's really expensive or when there's expensive hardware and hope but it doesn't work well in other cases where you have really fast moving implementation, so thank you, the best example is think of the building, mobile phone Apps smartphone Apps, like an iOS APP or an android APP I mean this code very easy to change very easy to deploy. And, to some extent the software is never done you release the APP and then you continually fix the APP update the APP revise the APP and so on, so in those environments having a heavyweight ceremonies is kind of restrictive. Because it forces you into doing things that require effort when really you just want to make the changes, you want to make and proceed and that's where it comes I

13p

How do agile processes deal with security well. There's nothing out there, like the Microsoft SDL for agile processes, the agile world, even though there is a considerable audience for agile processes. The agile community is not terribly tightly bound. There's no agile authority there's a group of researchers who put information out but they're not they don't have any real authority.

14p

There are research papers out there and approaches. That suggests that you can integrate security activities with agile methods. So there's a paper by Beznosov & Kruchten towards agile security assurance that basically breaks down different software assurance methods by general high level agile phases. So agile doesn't have a whole lot of structure to it in general terms. But they're generally is some idea that you're going to develop requirements do some kind of design typically that's an emergent design or it's just kind of an organic design that happens. And there's a big emphasis on implementation, when you iterate you do these activities over and over until the software just kind of emerges in existence and every step along the way you can do things like listed here. You can do things like internal reviews external reviews validation of requirements. Various types of testing security focused testing, which is an entire lecture in and of itself. And so there's ample opportunities to simply say you know it did every step along the way in our agile process we're going to do something to improve security.

15p

So let's take a look at scrum. Scrum is a very common agile process. It's really more of a management strategy. I know we talked about this just a little bit. That scrum it's not so much. I think of it as more of a structure for organizing meetings and assigning work. It's not so much it doesn't really go beyond that. You have different types of meetings like daily stand ups or sprint planning meetings or retrospectives when a sprint is complete you work in short bursts event time call sprints. And you focus on the common artifact called a product backlog that contains all of the information that you need to have about your product.

16p

So there are various flavors of scrum out there. Scrum is so abstractly defined that it's very easy to modify it to suit your needs. One such effort is called **Secure Scrum**. Secure Scrum basically bolts or tries to augment each part of scrum to better address security. So, for example, one of the phases of scrum is sprint planning. During sprint planning, developers get together and decide what features they're going to select from the product backlog and implement during the next development drive development. And according to Secure Scrum during that phase what you do is you identify all the security relevant aspects of the user stories. User story is the way that scrum write down requirements very lightweight approach and you make sure developers understand the implications of those security requirements.

17p

So, making sure they understand is still kind of a bit of work unto itself. Secure Scrum has this idea of **S-tags** and **S-marks**. So the S-tag, developer describes a security concern basically you just say for giving us your story. The user story template is more like as a particular type of user. I want to do X for the purposes of Y in my system. So it's very lightweight way to describe your requirement. So you'd mark that with an S-tag that would identify security relevant aspects. So here's an example, we talked about cross site scripting. So we would tag this a user story that says it could be vulnerable to a cross site scripting attack if it's not carefully implemented. That becomes a S-mark user story. So we want to buy by making this explicit in the backlog and doing it in a scrum like way with a S-tags and S-marks were able to you know we're able to better integrate security into our process.

18p

Another way we can adapt scrum to be more security focused is by changing the under the goals, remember, I said quality goals, goals in general really important to make explicit in a project. In scrum you have this concept of **definition of done**. Basically, the idea here is when you decide that a feature is completed, you must socialize that and come to a common understanding of what done actually means. This often means talking to the product owner or the client or. Because it's coming to a shared understanding, where everybody's in agreement that the future is indeed done it's a very organic again process.

19p

How to handle that with security. What you can do is you can expand the definition of done to include specific security activities. We will not call the feature done until we have completed fuzz testing, we've done a security focused review. Or we've run a certain battery of tools over the project over the feature and corrected all found issues. So this is a simple way that you can organize to just make sure that security is accounted for during a scrum project.

21p - Assessment

Our discussion about **security assessment**. This isn't so much assessing the product as it is the process. So once you've established a software development process to help foster and promote security it's important that you continue to evaluate that process. So this doesn't need to be terribly formal, although, as you see, there are frameworks that exist for this type of assessment. But it from time to time you just want to ask yourself are we doing a good job building secure software we're dealing with security and software, and this is kind of now, this can be kind of a tricky question. Because security is rarely the point of the project, the point is usually to set to implement some type of functionality. satisfy some type of set of user requirements that are rarely ever if that are rarely, if ever as simple as, build something secure it's kind of assumed and you're going to use the best security practices possible. So it's up to you the team to periodically look at your processes and ask yourself are we doing enough to promote security in the organization in the team.

22p - SSE-CMM (Systems Security Engineering Capability Maturity Model)

So there are a number of more or less maturity models. Out there I would be remiss if I didn't talk about the CMM style software maturity model for software security capability maturity models were developed at CMU as a way to assess the engineering capacity of an organization. So how well do they make software and we have a long history of doing this kind of work. So it's a good thing to see that certain maturity models have emerged to address software security specifically. So this **SSE-CMM** is one such model capability maturity model and it's broken down into more or less two areas.

23p - SSE-CMM: Organizational Process Areas

The security engineering aspects of the process and then there's various elements under that. So the organizational process defines an organization security engineering process makes it basically outlines and describes all the different things that you do in your organization LG or in your project team to facilitate developing secure software and then gives you the framework for doing an assessment of those areas.

24p - SSE-CMM: Project Process Areas

Going a little bit deeper. **The project process areas** are really focused on how specific projects are managed. So thinking about it like this, for every project if you are a mature organization, you have a set of activities that you must do to meet some organizational goal the SSE-CMM attempts to determine how well those activities are helping you to develop secure software. So we talked about capability we're not so much assessing individual outcomes of a project but we're looking at the strategies, the Organization has in place to deal with security in general.

25p - SSE-CMM Engineering Process Areas

So the **security** **engineering process areas** get really deep into the specific security practices that are evaluated. So doing threat analysis, threat modeling, doing risk analysis having elements in place to coordinate a security response. All of these different and so on and so forth, all of these different areas or looked at and evaluated to determine how well the organization, how well they are how good they are at dealing with security in a systemic way.

26p - **Software Assurance Maturity Model (SAMM)**

There are other similar approaches to software security assessment, one of them is the software assurance maturity model it's Open Source it's out there for you.

27p - SAMM (Cont.)

It assigns maturity levels, much like the traditional CMM to an organization. So organizations start at Level zero and go to level three. So there's four phases and of each step along the way you become your assessed as being a little bit better at managing security. If you're a face to you're really in a place where you're not just doing things to engineer secure software, but you're doing them well. we know at the beginning everything's kind of ad-hoc you're working a project kind of one off at a time, by the time you get to phase one to our higher level phases steps along the way, then you're able to articulate what specifically you're doing to promote security and really evaluating those security practices, for the sake of Improving the overall engineering process.

28p - Security Engineering Process: Assurance Process

The security engineering process is really relatively straightforward. You can think of it kind of as a circle where you identify risks identify have a process in place to make sure that risk mitigated and then having an engineering process in place to actually implement the system in a secure way. So starting from the Left year of the circle. You have employees assurance processes that make sure that security goals are explicitly articulated internalized and achieved so, what are you going to do to verify that you've actually you know completed

29p - Security Engineering Process: Engineering Process

identifying security goals you have an engineering process that takes those goals and makes them into software. So how do you take the goal and operationalize it in such a way that you can build a secure system

30p - Security Engineering Process: Risk Process

How do you assess the completed system do, what are your risk processes to help identify and mitigate certain types of risks.

31p - Software Security Framework (SSF)

There is another process, called the **SSF the software security framework** which extends SAMM the software assurance maturity model. There's basically a bunch of different practices that are organized around **four domains** **governance, intelligence, SDL touchpoints, and deployment.** And then each practice deals with specific security things security focused activities that must be done. so, for example in the SDL touch points you want to have some notion of code review that focuses on security, so we have security testing, obviously in deployment, which is post development, you want to have some notion of penetration testing or system level testing to actually evaluate the completed software.

32p - SSF Governance: Strategy and metrics, Compliance and policy, Training

The **governance** phase is most **focused on how you manage your security**. So this is what metrics are defined Howard post training done what types of things do you do to kind of keep the ship running in a good direction

33p - SSF Intelligence: Attack models, Security features and design, Standards and requirements

**Intelligence** **is a way to assess how well**. **The organization manages their own security knowledge**. So how do you, a theme of this class is going to be **learning to think like an adversary think like an attacker** that's all part of intelligence. So how do you what type of height when you do an assessment of your software for security. How do you approach that do you think like an attacker when you're doing that what types of security features and patterns you using development and other any kind of standards that are internalized for all projects. It's kind of security requirements handled in the system.

34p - SSF SDL Touchpoints: Architecture analysis, Code review, Security testing

So **SDL touchpoints** you already talked about a little bit, basically, how do you address. How do you what specific things do you do during software development to ensure that security is dealt with, we talked about code review and security testing there's also a contextual analysis over an entire lecture on auditing for software security

35p - SSF Deployment: Penetration testing, Software environment, Configuration management

Finally, **deployment** is what happens post completion rate, so this would be doing a penetration test or evaluating the software environment to understand. Not just how you know, looking at your system for how it handles specific threats is one thing, but once you deploy your system into the world now you have to make sure the world in your system play nice together.

36p - Using SSF

So the SSF is one of my favorites in terms of the assessment process or just a framework for thinking about how you how well, can you develop. Secure software because it's very specific, I can look at my process and say, are we doing an architecture security review, are we doing security testing and if not, why not, what metrics are defined to help evaluate security. See it makes things very explicit.

37p - Correctness by Construction (Reference - <http://rosaec.snu.ac.kr/meet/file/20090709d.pdf> 29p)

I want to finish talking just a little bit about. A specific type of software development lifecycle doesn't fit into anything like waterfall or agile. It's more like a general approach to developing software called a **correctness by construction**. This isn't new and I don't think it's widely used, but it is an example of what you can do if you really take if you make engineering high quality aka secure software, the priority then correctness by construction might be a process that works well for you.

38p - Correctness by Construction

So the idea is every step of development is rigorously, if not formally defined using formal methods formal specifications proofs and mathematical techniques, so that you can identify and remove defects at every step along the way, so this sounds very code oriented but you really apply this type of rigor to all phases of development. Right so requirements are translated into formal models, perhaps using was the formal specify a formal specification language, like the architecture specified using a formal a do architecture analysis definition language. Code is proved to be correct and verification,

39p - Correctness by Construction: 7 principles

There are seven principles that go along with correctness by construction. One of them is you **expect requirements to change**. My philosophy on this is requirements always evolve. I don't necessarily think they wholesale change often. You start requirements with concepts of what you want, and the real key is to elaborate and come to a shared understanding or rigorous understanding of what those requirements actually mean so that you can turn them into software. So in correctness by construction, the idea is that **requirements are incrementally developed and rigorously verified**. So that as they evolve, you can reduce that you can more quickly converge on a standard or common agreement for what the requirement is. **Knowing what you're testing**. The important thing here is the testing serves multiple purposes. So one form of **testing is to actually make sure it works (=Build, Debugging)**, in another form is to **show that it works properly (=Correct, Verification)**. You often hear about testing for functional correctness, I think that is kind of bare bones minimum what you do during software development. And the second approach would be to or actually go bug hunting. Identify defects in the implementation. So that it's a slightly different goal than just proving something works as you expect we all make test cases to show that, when I run it with a certain when I run my software, with a certain input I get the identical that required output. And the harder type of testing is what happens, how can I, when I give it input that I don't expect. We'll see this in a completely different lecture. Another principle is to **eliminate errors before testing**, so a core idea behind curtains back instructions to do things, before you get to testing to make testing easier. Testing can be kind of expensive. It might be easier to run static analysis tools or do a code review or do somewhere or build or verify a model of your system. Certainly, will be cheaper.

40p - Correctness by Construction: 7 principles (Cont.)

Other principles are more practical. So **write software that is easy to verify.** Keep things simple. This has various different definitions of what simple means, but more or less I think if it is the software starting to look really complex maybe refactoring is necessary. **Don't do things all at once (=Develop incrementally)**, make changes make small changes and re-verify except the fact that this is a difficult task. **Some aspects of software development is difficult**. And, recognize that software would more than likely fit into a larger context. So, the code itself is important, but there are other aspects of the system that must be dealt with **(=Software is not useful by itself)**, for example, user manuals documentation, well commented code, well written test cases that are rigorously maintained all of that is important. There's more to correctness by construction, but for the sake of our purposes, I think the seven principles mean even if you don't adopt correctness by construction, because it is really an arduous process to implement, I think the seven principles work well for developing high quality and secure software.

41p - Microsoft’s Modern Engineering

Microsoft is a big player in the secure software development space and we've already looked at their SDL. They have another effort to push security called modern security from modern engineering came out in 2016

42p - Microsoft approach

The approach is one that will cover a lot in this course, because I think it's becoming prevalent in industry. So the idea is to push as much into automation as possible. then embrace the DevOps culture, when I say DevOps culture, I mean this environment where we're continuously running tools or making commits or updating code repositories automatically making builds automatically running a battery of tests automatically are encountering events that trigger some type of automated process. The idea behind the Microsoft approaches to put as much security control as possible into these really short release cycles. So this is difficult for security because security often slows things down by having really a lot of checkpoints along the way.

43p - Continuous assurance: Phase - Auditing and logging, Authentication, Authorization, Application business logic, Configuration setting, Cryptography, Data handling, Input validation, Output encoding

So the idea would be security, something we call **continuous assurance** and you can see, the different phases here. What we want to do is make security kind of inseparable from a development. We do that by instrumented our processes to check for security thing, so this might be as simple as running a battery of security testing doing automated fuzz testing for every commit or making a part of a check in date more or less. But I think that it's a good strategy and I'm a big fan of continuous integration strategies, I think they work well, I think they allow us to find problems faster than using other approaches and so I fully support pushing as much of the security evaluation process into a continuous development or a continuous integration approach.

45p - Risk Management

Okay let's finish the day or finished the lecture by talking about **risk management** and **conducting an effective risk assessment** for a project. So this is not to be confused with the threat model which is identifying specific things that pose a threat to your system and for the purposes of countering them. **Risk management is about identifying things that put the project**, rather than the product at risk, although those can often be one in the same. So there are projects or there are risks that are associated with personnel problems or lack of risk associated with deficiencies in home in project management. And the idea is that we want to aggressively and actively try to anticipate and mitigate risks.

46p - Risk tolerance and security

**Risk tolerance** is kind of an inseparable concept from security. So, then the question is really simple. How much security is enough? When we go through threat modeling, we'll look at defending against specific types of threats. But kind of framing that process is how much you're willing to invest in mitigating threats and mitigating risks. To answer this question, you need to think about your project in terms of its assets and that goes beyond just the physical assets, that is, the business processes in place to achieve organizational goals. Under what conditions are those assets put it risks. And then you have to understand to figure out for the level of risk that you identify what we do to address those risks.

47p - Risk Process

So the basic process here is to the **risk management process** is to number one actively assess the risks folk facing your project. So what are the things that you concern yourself with that are threatening to the project success. After **you identify the risks**, you should figure out how you're going to **address each risk** and then **control the risk** by mitigating it one way or another.

48p - Risk management framework for security

So, again security is kind of inseparable from risk management. So we want to identify risks that have security flavors. Here's the basic framework for that. There are more or less **five steps**

49p - Step 1. Understand the business context

In step one, we **understand the business context**, and so, why are we here what is how, what is the goals of our organization in our project, what is our missions. We want to type business goals to mission. This can be, for example, maintaining service availability, so if that's a critical asset, **a business goal**, then it makes much easier to identify threats to that goal or identify risks that can affect that goal, for example, denial of service.

50p - Step 2. Identify business and technical risks

That's that comes in step two right in step two we sit down and think about risks. So the more or less there are a product and process risks another way to think about that is there are **business risks**, which are threats in the achieving our business goals. Financial loss, damage to reputation, you can imagine, security compromises. High profile security incidents can certainly need to those unfortunate outcomes. There are **technical risks**, which are more like product risks. So things that can damage the product great. This can be a high profile defect security concern reliability problems or ineffective or insufficient functionality. The process of coming up with risks, so the idea here during risk management is to identify come up with a list of risks great. We'll call that the **risk register** we'll get to the format of that in a moment, but you identify the set of risks that you think are most important to deal with. Prioritize those risks and then take steps to mitigate them. And those steps might be perceived process oriented. So it's not like you just immediately turn the risk off. What you do is you think about this is something that should occur during the project can be very detrimental. So what can we do to ensure that risk never transpires never becomes a problem. Risks aren't things that happen, they are things that may happen that have negative outcomes.

51p - Risk format

So let's talk about what a good risk looks like. The risk has two parts of very condition and consequence. So the important thing here is that the condition is a statement of fact. It's not a may happen or what if question it's this event occurs this thing transpires. And then the consequence is some notion of loss. Security vulnerability reported is could be a risk. And the consequence would be. Well there's a number of consequences. There are consequences would be add research competitors or loss of product revenue or loss of revenue, because we had to deal with the issue something like that I'm kind of brainstorming, which is what we'll do I'm going to have our live exercises interested.

52p - Risk Register

So I said, you come up with a list of risks. So here's an example of a condition, consequence format. So ‘communication with remote employees difficult’ something we've all had it ‘prob’ like we do with it during the pandemic. The consequences critical software components not or elements not completed in time and then we write down, we can capture as many verses we can what's a good number of risks. there's no hard and fast rule i've heard about a dozen risks it becomes kind of a fruitless endeavor if you have more than a dozen risks on a project decides, for example, of our course project, then you have more fundamental problems, you need to go back and evaluate so i'd say capturing a top 10 Number first would be is important.

53p - Step 3. Prioritize risks

The third step, **prioritizing the risks.** So again, there are no hard and fast rules. The prioritization scheme, I have here is the **prob**ability \* **imp**act. So **probability on a scale of 1 to 5**, with 5 being most likely and 1 being the least likely. And then **impact again 1 to 5** with 5 being severe impact and being limited impact multiply those together and you get a reading. And that's a convenient way to do the prioritization if all the rating attends. Everything is for whatever everything is high probability, high impact or I guess the maximum will be 25 high probability, high impact and you've got other problems you have to deal with the risks are considerable.

54p - Step 4. Mitigation

And then the last step four is to **identify mitigation.** So if you identify a risk and you think it is a something you should attend to well. Then exactly how are you going to deal with it. So going back to the remote employee example here. if I think this is a risk to the project, that I should take steps now to make sure that to mitigate that risk maybe I have weekly status calls or I as a team manager would calling the individual remote individuals, just to see how they're doing to make them feel connected and engaged during these difficult times. The second risk here is more product oriented risk, where we have specialized hardware, so you guys certainly have been at the hardware is damaged. One way to mitigate that would be to dedicate a specific role, a safety engineer or something to handle all hardware be the focal point of hardware issues.

55p - Step 5. Validation

And, then the last part here is to **execute the strategy**. So very often risk meant to SEC teams conduct a risk assessment. And then just discard it they do it for the as like an academic exercise, but really it's something you should be done, that should be done all the time, so make risk management part of your core process, maybe when you do sprint planning or planning for whatever an iteration. You review the risks or when an iteration completes and you're doing a post mortem on how things went. You update the risk register, but you make risk kind of core part of your process. You identify new risks at different phases of the project and you mitigate you make sure everybody understands their role in risk mitigation. That is continuous risk management, and that is, I think the most effective way to conduct it.

56p

So, in summary, for the project lecture, security needs to permeate all phases of your software development process. This doesn't need to be hard you just have to ask yourself for every whatever process you select for every activity prescribed by that process, what can we do to promote secure implementation or security in the project. Agile processes tend to focus on communication and coordination between people, so you want to socialize security controls. Traditional processes are more ceremonial and have communication through artifacts so certainly those artifacts in the do address security concerns. Risk management is a core part of any process where you identify concerns and risks for the project as we'll do a live exercise with on risk management, we will develop risks you think will need to be addressed during the course, so thank you for your time and I will see you next time.