**Secure Software Systems  
Project Final Report**

**Abstract:**

The increasing use of software for essential functions has raised concerns about software security. Sadly, a lot of software development methodologies overlook security, which exposes software to intrusions. Secure Software Engineering (SSE) seeks to address this by placing security at the forefront of the software development process from start to finish.   
  
This research examines several security models, techniques, and best practices to determine what functions well and poorly. It emphasizes how crucial it is to consider security from the outset, recognize potential threats, and test software to ensure security. The report also looks at the particular difficulties in protecting the Internet of Things (IoT), particularly in smart grids and cities.

**Introduction:**

This research focuses on protecting software systems from unknown dangers such as malevolent hackers. It highlights how software is becoming more and more important in everyday life in a variety of ways, from communication and financial transactions to the storage of important data. The extent to which software can be abused and exploited increases with its integration into society's infrastructure. Software system vulnerabilities can result in serious consequences for society, including the loss of confidential data and monetary losses. According to this study, in order to successfully reduce these risks, software frameworks must have strong security measures in place. In digital ecosystems, trust and integrity are crucial and are inextricably linked to software security.

The study essentially highlights the pressing need for all-encompassing approaches to strengthen software security and reduce potential vulnerabilities in a digital world that is becoming more interconnected by the day.

**Background:**

Following is a comprehensive review of fifteen papers, we have gone through in order to gain knowledge to summarize the best software practices and evaluate them.

**Paper 1:** [1]

The study talks about Secure Software Engineering (SSE), which is basically about building software that can handle attacks and keep our information safe. It's like making sure the doors and windows of a house are strong enough to keep burglars out. But the study found that many methods used to build software don't focus enough on security, which can leave them vulnerable to attacks.

To fix this, the study wants to look at all the research out there about how to make software more secure. They're going to search through a bunch of academic papers to see what methods and techniques are being used to make software safer. They'll also look at where these papers are being published and what kind of research is missing.

Overall, the goal is to understand the current state of software security, find areas where more research is needed, and give guidance for future studies to make software development safer for everyone.

**Paper 2:** [2]

Here are some of the most widely used security models.

Software Assurance Model(SAM):is adaptable to all scenarios for the global clients, is developed from 11 development models nd analysis of literature review. Three case studies on software development is done.

Secure software engineering (SSE) involves every step of creating software with security means designing, building, and testing the software in a way that makes it secure from the start. It consists of implementing secure software development life cycle (SDLC) processes and using secure software development (SSD) methods. By integrating security measures throughout the entire software development process, we can create software that's much less vulnerable to attacks and keeps our data safe.

Software security is like the software's defense system. It's about how well the software can handle and recover from attacks that try to mess with it on purpose. Just like how our immune system fights off germs to keep us healthy, software security protects the software from bad stuff that could harm it or the information it holds. When software has good security, it's less likely to get hurt by those attacks, and it can keep working reliably.

Secure software development process:

1.Identifying potential abuses, defining security requirements, analyzing architectural risks, reviewing and fixing code, conducting penetration testing, and managing security operations.

2. Team Software Process (TSP) specifically to help software teams become high-performing and produce excellent results in terms of security.

3. Creating a method that evaluates device assets, analyzes risks, and defines weaknesses, threats, and risks to ensure appropriate security measures are in place.

4. Introducing security paradigm for agile methodology in web application development, extending security practices to overcome specific challenges in this context.

5. Developing Secure Software Development Model (SSDM), which provides training to everyone involved in software development to ensure they have the necessary security knowledge and skills.

6. Designing the Secure Software Design Maturity Model (SSDMM) to help measure the maturity level of software development organizations in terms of security practices.

Each of these approaches offers different methods and frameworks to integrate security into the software development process effectively.

CMMI is a model for improving development processes and product quality. It's used to measure Inner Source implementation maturity, which adopts open-source practices internally. Additionally, a Software Maintenance Maturity Model (SMmm) based on CMMI helps assess and enhance the quality of software maintenance activities within organizations.

After a lot of research in this feild experts made a model called SAM to help companies improve their software security. They tested SAM with real companies and found that it worked well and was easy to use, making software safer for everyone.

Along with that we also need to provide security for IoT systems too as there is increase in use of IoT systems.

**Paper 3:** [3]

Modernising power infrastructure requires the use of the Internet of Things (IoT)-powered smart grid. However, because of its complexity and dependency on communication networks, it is susceptible to cyberattacks, endangering both public safety and national security. In order to improve smart grid security, a great deal of research in industry, government, and academia has been conducted for assessing cyberthreats and suggesting measures. It looks at different kinds of attacks, weaknesses, and defences against them with the goal of expanding knowledge and directing future research on smart grid application security.

**Paper 4:** [4]

Software Engineers should be taught about the security methodologies while developing inorder to ensure security for software and protect it from unauthorized access.

Best Security Practices:

Setting Security Standards:It's important to lay down rules for security, like training staff on critical issues and following 21 security rules to keep software safe from unauthorized access and infections.

Practicing Secure Development: During the software design phase, we need to think about security from the start. This involves creating a plan that considers potential risks, documents assumptions, and involves external reviewers.

Writing Secure Code: When writing the actual code, we should use tools to check for common vulnerabilities and ensure that the code is free of errors.

Testing for Security:We need to test the software to make sure it's secure. This includes checking if security features work properly and testing for potential attacks based on known patterns.

Learning from Attacks:It's important to learn from past attacks and update our security measures accordingly.

Understanding Threats: We need to identify potential threats early in the development process to protect the software from vulnerabilities and attacks.

Analyzing Risks: By analyzing risks at different levels of the software system, we can better understand where vulnerabilities might exist.

Using Attack Analysis Tools: We can use specialized tools to analyze potential attacks and vulnerabilities, helping us design more secure software.

By following these steps, we can create software that's not only functional but also safe from cyber threats.

**Paper 5:** [5]

IoT devices are used mostly now a days to improve our lives. There are several challenges which are found specifically in IoT systems.

IoT devices communication happen in the wireless network and we can operate them remotely Example checking glucose level of the paitent. Industries use IoT devices to easy tasks and Iot devices are mostly prone to security threads because of Wireless Sensor Networks

(WSNs), Machine-to-Machine (M2M) or Cyber-Physical

System (CPS). Iot devices are interconnected with limited power and memory. Mostly encryption is used i.e encryption algorithms are used to secure the data for these constraints.

Techniques like secure multiparty computation and attribute-based access control are explored to ensure privacy for IoT users. Security issues in cloud-based IoT, such as identity and location privacy, are also being addressed. Survey studies have highlighted security threats in domains like smart homes, smart cities, healthcare IoT, and industrial automation. Despite these efforts, there's still a lack of empirical investigations into the challenging factors of IoT security, which this study aims to address.

**Paper 6:** [6]

To make software systems secure from the very beginning of their development the process is called Security Requirements Engineering. Security Requirement Engineering is about building security into software development from the beginning. There hasn't been enough research on this as the only suggestion is to start security practices from the development phase. Experts are conducting detailed reviews to make software safer in the future for better out come in security development.

**Paper 7:** [7]

The DIMACS Software Security Workshop delved into various aspects like security engineering, architecture risks, and analyzing security. It covered topics such as mobile security and education. The workshop aimed to bridge computer security, programming languages, and software engineering to advance software security.The DIMACS Software Security Workshop is a gathering where experts in computer security and software engineering come together to discuss important issues in making software safer. They talk about things like security risks, analyzing security, and how to train people in security. The goal is to share knowledge, find new ideas, and work together to make software more secure. By bringing different communities closer, like computer security and software engineering, they hope to find better solutions to problems. The workshop helps researchers and practitioners share what they've learned and figure out what to focus on next to keep our software safe from hackers.

**Paper 8:** [8]

The challenges faced in implementing DevOps, especially regarding security. Through a systematic review and expert surveys, 18 key security challenges were identified. Using a method called PROMETHEE-II, these challenges were prioritized to create a taxonomy. This helps practitioners better address security issues in DevOps, ensuring smoother and safer software development processes. This approach, although new to DevOps, has been successful in other fields like medicine and banking.

DevOps, a way of organizing software development teams, is becoming really important. It helps teams deliver software faster and more reliably. But there's a problem: making sure everything is secure. This study looks at what those security problems are and how to fix them. They did a big review of what experts have said about DevOps security, then asked more experts about it, and finally used a special method called PROMETHEE II to figure out which security issues are the most important to tackle first.

**Paper 9:** [9]

Authentication techniques in computer security, exploring both single-factor and multi-factor methods by combining these techniques for enhanced security and examine frameworks for comparing and selecting authentication methods. Through a systematic literature review, the article identifies commonly used authentication techniques and discusses criteria for comparing them. It concludes by emphasizing the need for further research in this area. Overall, it offers valuable insights for industry professionals tasked with selecting authentication methods for various applications.SLR is used to find the best authentication schemes for the applications and also used to compare the existing systems.

**Paper 10:** [10]

There is increasing in growth in software industry and there are different ways to evaluate SDLC security after experts analyzing the papers and conducting research many studies focus on certain parts of creating and improving things, like putting ideas into action, making changes over time, and getting feedback. But areas like testing don't get as much attention. So, it's suggested to stick to established guidelines when doing case studies to make them easier to repeat and more like real-life situations. While some parts of the process have been looked at a lot, there's still more to explore in other areas. The goal is to have a smoother and more flexible process from the start of an idea to testing it out and getting feedback.

**Paper 11:** [11]

Traditional approaches often overlook security until after system definition, leading to vulnerabilities. Integrating security with software engineering is proposed to address this. The paper advocates for a methodology that embeds security throughout system development, offering an approach exemplified by a health and social care information system, enhancing existing practices. As traditional practices are the known for there standard impact on the security systems.

**Paper 12:** [12]

By focusing on the concern of exploiting vulnerabilities in software development. Experts had conducted an review on 867 papers by highlighting key findings across various stages: Requirements, Design, Construction, and Testing Security. Common themes include Misuse Cases, Component-based development, Threat Models, Static Code Analysis, Vulnerability Detection, Vulnerability Scanning, and Penetration Testing.

It concludes that diverse methodologies, models, and tools exist for each stage of secure software development which will help developers to develop security from the early stages of the SDLC process. As we go through various iterations in SDLC the development in the security phase also improves parallelly as per the requirement.

**Paper 13:** [13]

New Approach also there is a new approach of developing securing model using a technique of Model Driven Security (MDS).

Here are the some of the steps and roles should be followed to create a secure software

Designers has to determine how the system should work and what security it needs. Then, tools automatically create the system's structure, including how access is controlled.

Flexible Language Instead of sticking to one way of describing systems, they suggest using a mix of languages for system design and security.

Need to show how to blend different UML languages with a security language to make sure access control is clear.

how the system configures server-based apps' access controls automatically. Both declaring rules and coding rules for access are covered.

Basis Their approach builds on an established technique known as role-based access control and is founded on a strong understanding of how access control functions.

Use in Practice A UML-based tool that assists designers in implementing this technique in actual projects has been used by them to put this into effect.

Testing To ensure that this strategy is effective, they have tested it and shared the test results.

**Paper 14:** [14]

Even after developing software there will be security issues in the phase of maintenance also. Sometimes software might change due to any change in requirements.Due to these changes where might be changes in the security and some flaws might raise in software.In order to develop a good secure application need to focus on the maintainability feature and designing software for maintainability to check for the issues easily.

Its developed in 3 phases:

1. Initial phase:Failures were categorized into functionality and safety issues. Only 171 out of 848 identified anomalies directly caused application failures due to poor maintainability controls. Mitigation strategies were devised for each phase of the SDLC.A software application runs smoothly and doesn't break down often, you can't just throw more resources at it or do maintenance more often. You also need to follow good practices during every step of building the software good practices reduced the number of times the software failed. This means that the software became more reliable.

MSDLC Model is used to put the all the good practices and keep track of the problems and fix them and continue maintainability by improving the quality of the software and improving features.

Overall, using this model can help make sure that the software doesn't have as many problems and doesn't break down as often, keeping it running smoothly for longer periods without any major issues.

**Paper 15:** [15]

By expanding number of linked devices in the Internet of Things (IoT). It presents the "lightweight named object" concept, which uses Information Centric Networking (ICN) to streamline device management and interaction.

The lightweight named object solution gives physical IoT devices in the ICN namespace distinct names. This abstraction provides various advantages:

Increased Programming Simplicity: Without the need for extensive network setups, developers can focus on device functionality.

Extended Functionality: The abstraction enables faster software updates and the inclusion of new functionality to devices.

Common Interaction Logic: Devices can communicate with one another using their allocated names, resulting in a more managed IoT environment.

The research uses proof-of-concept implementations and performance evaluations to establish the practicality of this technique, implying that it has the potential to support future large-scale IoT deployments.

Imagine millions of smart devices talking to each other – that's the future of the Internet of Things (IoT). But managing all these devices can get messy. This paper proposes a clever solution: giving each device a unique name, like a nickname.

Here's the gist:

With so many devices, keeping track of them and how they interact can be a nightmare.

This paper suggests using a special network system (ICN) to assign unique names to devices, making them easier to manage and control.

Think of it like giving each device a clear, unique nickname within the network. This simplifies programming and lets devices easily interact with each other.

Plus, devices can be updated and improved without physically touching them, just like updating an app on your phone.

This is a promising approach for handling the explosion of connected devices, making the future of IoT smoother and more manageable.

**Table 1: Summary of the Identified Publications and Presentations**

|  |  |  |
| --- | --- | --- |
| No | Citation | Description |
| S1 | [1] | The paper indicates discrepancy between the required security measures and the current development methods is brought to light by the Secure Software Engineering (SSE) study. Through a thorough analysis of numerous academic papers, it seeks to evaluate procedures, pinpoint weaknesses, and improve security protocols. |
| S2 | [2] | The paper indicates the most widely used security models like SAM and SSE along with their importance and use of IoT systems in integrating security measures and mitigating the vulnerabilities in Software Development Lifecycle. |
| S3 | [3] | The paper addresses the various security threats in the IoT domains, and their vulnerability to cyberattacks due their limited sources and advocates investigations into IoT security challenges. |
| S4 | [4] | The paper highlights implementing various security methodologies into software engineering to ensure software security and other best practices to set the security standards to analyze the risks and threats and develop a secure software. |
| S5 | [5] | The paper highlights current privacy-focused measures like access control and encryption, this study highlights the need for empirical research in addressing the security challenges posed by Internet of Things devices. |
| S6 | [6] | The paper indicates the importance of Security Requirements Engineering for security development. |
| S7 | [7] | The paper highlights the DIMACS Software Security Workshops to enhance the software security from risks through analysis and training. |
| S8 | [8] | The paper highlights the security challenges in implementing DevOps through systematic reviews and expert surveys. |
| S9 | [9] | The paper showcases single-factor, multi-factor authentication techniques to enhance security through authentication methos selection and comparison. |
| S10 | [10] | The paper focus on evaluating SDLC security to ensure smoother and more adaptable software development process. |
| S11 | [11] | The paper suggests embedding security in the software engineering processes to enhance overall security practices. |
| S12 | [12] | The paper indicates the significance of addressing the vulnerabilities in software development stages. |
| S13 | [13] | The paper highlights the use of Model Driven Security (MDS), for testing to validate the efficiency of the software. |
| S14 | [14] | The paper highlights the necessity of addressing the security issues and proposing the use of MSDLC model to improve software quality. |
| S15 | [15] | This paper showcases the approach for handling the explosions of connected devices making the IoT smoother and more manageable in future. |

**Research Method:**

Research method for studying software security and Secure Software Engineering (SSE):

Problem: We Identified the problem of increasing software risks in critical domains especially IoT.

*Objectives:*

Evaluated existing software security models. We Identifed widely used security models.

Explore challenges and opportunities in integrating security into software development.

Studied the impact of emerging technologies on software security.

Research Design: We Use a mixed-methods approach with a structured framework.

Data Collection: Gathered academic papers, industry reports, and case studies. Considered interviews or surveys.

Data Analysis: Analyze qualitatively and quantitatively, comparing and contrasting different models.

Findings: We presented key findings, interpreted them, and discuss implications for practice, policy, and future research.

Conclusion: Summarize findings, provide recommendations, and suggest areas for further research.

**Results:**

From our by analyzing the papers we understood the importance of software security in our daily lives, as protecting our digital assets is essential. They discuss Secure Software Engineering (SSE) methods, like the Software Assurance Model (SAM) and Secure Software Development Model (SSDM), which help make software safer by integrating security measures throughout development.

We understood the challenges of IoT devices, such as smart cities and healthcare devices, requiring strong security due to increased vulnerabilities in the wireless network.

Education and training for software engineers are vital, with a focus on teaching security best practices early on, like setting standards, coding securely, and learning from past breaches.

Authentication, DevOps security, and integrating security into software engineering are also discussed, emphasizing the need for proactive security approaches.

Finally, the papers we have referred highlighted the importance of prioritizing software security, suggesting areas for further research and offering guidance to improve software development practices, ensuring software systems' integrity and reducing risks.

**Discussion and Challenges:**

Here we took an IOT device like smart cities here some of the challenges :

Smart Cities like: smart trash collection and crime management etc.... have many vulnerabilities to the internet.

Privacy leakage in VR and IoT occurs when sensitive data is exposed during communication or storage. It's measured by assessing encryption, data sharing policies, and IoT device security. Conducting Privacy Impact Assessments ensures compliance with regulations and identifies and mitigates risks to protect user privacy.

AI is super important nowadays, but it can also be used by bad actors to find out private stuff. For instance, companies might use AI to learn about us through our devices. Hackers are getting smart too, learning how to mess with AI systems to get what they want.Regular methods like spotting intrusions aren't enough for complex IoT networks. We need simpler ways to find and predict attacks. Plus, bad devices sneaking in could steal info or mess with others. Handling lots of data from IoT devices is tough too. Smart homes face similar problems, like keeping our info safe and stopping hackers from messing with our devices. And for healthcare devices, we need to make sure they're secure and can't be messed with by hackers. Trust is key too, making sure we can rely on these devices.

The Internet of Things (IoT) makes life easier by sharing data seamlessly through smart devices. However, security and privacy are big worries. We've identified 21 challenges, like AI's impact and data tampering. Experts ranked these challenges, helping us understand what's most important. We're working to find solutions and make IoT safer.

**Conclusion:**

In conclusion, we evaluated how important software security is becoming in an increasingly digital society. Since software affects many facets of life, it is impossible to overlook the risks posed by cyber threats.

One such strategy is Secure Software Engineering (SSE), which emphasizes including securitycontrols at every stage of software development. Software can be made safer by specifying security requirements, identifying risks, and doing extensive testing.As we are using IoT for making day to day life better Internet of Things (IoT) call for creative solutions. In the Internet of Things, access control, secure multiparty computation, and encryption can all assist safeguard privacy and lower risks.Addressing growing risks requires cooperation between professionals in computer security and software engineering, such as that found at the DIMACS Software Security Workshop.

Software security faces both opportunities and challenges as approaches like DevOps develop, and these must be handled skillfully.Enhancing software security is influenced by ideas like lightweight named objects in the Internet of Things, maintenance, and authentication.To sum up, it is critical that software development processes give security top priority. Through the utilization of current research, identification of research gaps, and promotion of cross-disciplinary collaboration, we may construct a future in which software is safe and resistant to cyberattacks, guaranteeing the security and integrity of digital ecosystems.

**References:**

[1] G. McGraw, ‘‘Six tech trends impacting software security,’’ Computer, vol. 50, no. 5, pp.  
100–102, May 2017.

[2] J. C. S. Nunez, A. C. Lindo, and P. G. Rodriguez, ‘‘A preventive secure software  
development model for a software factory: A case study,’’ IEEE Access, vol. 8, pp.  
77653–77665, 2020.

[3] M. Z. Gunduz and R. Das, ‘‘Cyber-security on smart grid: Threats and potential  
solutions,’’ Comput. Netw., vol. 169, Mar. 2020, Art. no. 107094.

[4] A. Sharma and M. P. Kumar, ‘‘Aspects of enhancing security in software development life  
cycle,’’ Adv. Comput. Sci. Technol., vol. 10, no. 2, pp. 203–210, 2017.

[5] M. A. Akbar, A. Alsanad, S. Mahmood, and A. Alothaim, ‘‘A multicriteria decision  
making taxonomy of IoT security challenging factors,’’ IEEE Access, vol. 9, pp.  
128841–128861, 2021.

[6] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, ‘‘A systematic review of  
security requirements engineering,’’ Comput. Standards Interfaces, vol. 32, no. 4, pp.  
153–165, 2010.

[7] G. McGraw, ‘‘From the ground up: The DIMACS software security workshop,’’ IEEE  
Secur. Privacy, vol. 1, no. 2, pp. 59–66, Mar. 2003.

[8] S. Rafi, W. Yu, M. A. Akbar, A. Alsanad, and A. Gumaei, ‘‘Prioritization based taxonomy  
of DevOps security challenges using PROMETHEE,’’ IEEE Access, vol. 8, pp.  
105426–105446, 2020.

[9] I. Velásquez, A. Caro, and A. Rodríguez, ‘‘Authentication schemes and methods: A  
systematic literature review,’’ Inf. Softw. Technol., vol. 94, pp. 30–37, Feb. 2018.

[10] A. S. Guinea, G. Nain, and Y. L. Traon, ‘‘A systematic review on the engineering of  
software for ubiquitous systems,’’ J. Syst. Softw., vol. 118, pp. 251–276, Aug. 2016.

[11] H. Mouratidis, P. Giorgini, and G. Manson, ‘‘When security meets software engineering:  
A case of modelling secure information systems,’’ Inf. Syst., vol. 30, no. 8, pp. 609–629, Dec.  
2005.  
[12] H. Nina, J. A. Pow-Sang, and M. Villavicencio, ‘‘Systematic mapping of the literature on  
secure software development,’’ IEEE Access, vol. 9, pp. 36852–36867, 2021.

[13] D. Basin, J. Doser, and T. Lodderstedt, ‘‘Model driven security: From UML models to  
access control infrastructures,’’ ACM Trans. Softw. Eng. Methodol., vol. 15, no. 1, pp. 39–91,  
2006.

[14] V. Suburayan, ‘‘Software development life cycle model to improve maintainability of  
software applications,’’ in Proc. 4th Int. Conf. Adv. Comput. Commun., 2018, pp. 270–273.

[15] L. Bracciale, P. Loreti, A. Detti, R. Paolillo, and N. B. Melazzi, ‘‘Lightweight named  
object: An ICN-based abstraction for IoT device programming and management,’’ IEEE  
Internet Things J., vol. 6, no. 3, pp. 5029–5039, Jun. 2019.