Security Requirements Analysis For The Development Of Secure GEOSCHEMACS

K. V. Maruthi Prasad, J. Krishna Kishore
ISRO Satellite centre, HAL Airport road, Bangalore-17, India

Abstract

This paper titled "Security Requirements Analysis for the Development of Secure GEOSCHEMACS" is an insight into the software security requirements of GEOSCHEMACS (ISRO's in-house developed set of software components used for Indian geostationary satellite health monitoring, analysis and ground control). This analysis is part of the process of introduction of building security in Software Development Life Cycle of GEOSCHEMACS under the research topic "Secure Software for Indian Spacecraft Ground Software Elements, GEOSCHEMACS". The primary objective is to identify and list software security requirements with reference to secure GEOSCHEMACS system development. It is focused at tabulating the software security requirements for avoiding most of the programming language inherent vulnerabilities.

Keywords*:* requirements analysis, software security, mission ground software, threat, operational environment.

I. INTRODUCTION.

ISRO (Indian Space Research Organisation) is the premier government institute involved in space research and development activities. ISRO has been known for it's accomplishments in nation building through science & technological innovations in space field. GEOSCHEMACS (GEOstationary SpaCecraft HEalth Monitoring Analysis and Control Software) is the in-house developed end to end software solution and primary set of ground software elements used for Indian geo mission health monitoring, control and analysis. GEOSCHEMACS is a software package based on client / server architecture with the development environment primarily consisting of C/C++, X/Motif, Oracle on UNIX / LINUX Operating System flavours. It is enriched with web version of spacecraft health monitoring and analysis. The total size of GEOSCHEMACS is around one million lines of source code. The role of ground software elements has been crucial and critical in meeting the ever expanding space services for users. It is necessary to evolve secure software for ground elements used for spacecraft health monitoring, analysis and control so that there is no disturbance in supporting space services.

Secure Software is the idea of engineering software so that it continues to function correctly under malicious attack. The inability of a system to perform functions without violating an implicit or explicit security policy can be taken as an attack. An instance of a fault in the specification, development or configuration of software such that it's execution can violate an implicit or explicit security policy is called as the vulnerability. A security vulnerability is defined as a flaw or weakness (in the case of software bug) that can be exploited by a threat (an attacker or malware) to cause harm or damage. Not every software security vulnerability is exploitable, but every security vulnerability can cause damage. Vulnerability can cause software to hang or crash, may allow for privilege escalation on the system, can cause software to act in unintended way or even allow for execution of arbitrary code. The amount of time, money and human resources involved in recovering from the damage has been enormous.

With the advent of high technological operational environment in the mission control centre, various security issues can arise which includes insider threat attack. Whatever may be the scenario these interruptions or crashes or attacks through vulnerabilities may cause disturbance in spacecraft health monitoring & control and analysis. This in turn cause disturbance to the space services for various customers; hence direct damage of money and reputation.

Even though GEOSCHEMACS software development process has been established at IEEE 12207 international standards, the concept of secure software development in the SDLC (Software Development Life Cycle) is yet to be introduced. The main reason of many bugs has been buffer overflow (one of the security vulnerabilities). Hence, the necessity of ensuring the software is constructed secure has been at higher priority. The only way of doing this is to integrate a security mindset & process throughout the SDLC. With software security aspect into consideration a typical secure software development life cycle can be depicted as in Figure 2:

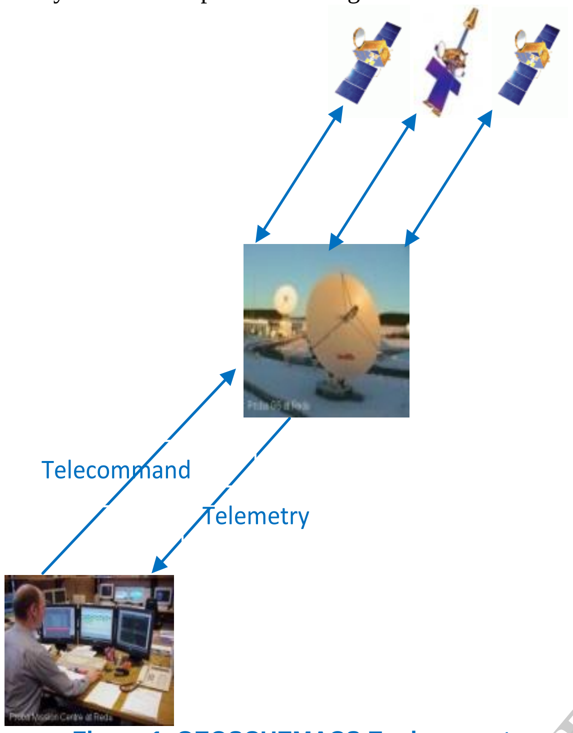

Figure 1. GEOSCHEMACS Environment

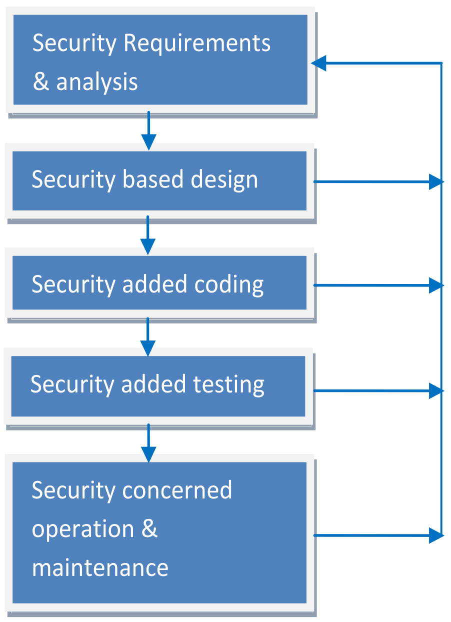
Security Requirements & analysis
Security based design
Security added coding
Security added testing
Security concerned operation & maintenance

Figure 2. Secure Software Development Life Cycle

GEOSCHEMACS has been evolving with the incorporation of new features & requirements and is a package of complex nature. Complexity issues and language inherent vulnerabilities are the primary causes of insecure software. Awareness and understanding about the necessity of secure software development process is one of the major steps required to be attended. The first phase of any software life cycle is requirements engineering which contains elicitation, analysis and validation of the requirements. Secure software development life cycle contains requirements analysis having attention & concern related to software security requirements. As part of building security in GEOSCHEMACS software development process and as part of the research work being carried on the topic of "Secure Software for Indian Spacecraft ground software, GEOSCHEMACS", this paper presents the preliminary & first cut security requirements engineering for GEOSCHEMACS.

II. The Approach

The approach for security requirements engineering of GEOSCHEMACS shall be a solution primarily for identifying current vulnerabilities that could produce exploitable software. The process of producing software security requirements is complicated with some issues which increase the difficulty of producing such requirements. The development lifecycle can effectively reduce / prevent vulnerabilities only when these issues are overcome. The issues include [2]:

1) Constant changing of security vulnerabilities
2) Difficulty in validating the security requirements stated in negative tone
3) Language and platform dependent software security requirements definitions
4) Possibility of testability & verifiability of security requirements throughout the phases of software life cycle
5) Process of selecting the required security requirements for the application or system with reference to all the software security requirements covering security vulnerabilities to cryptography
6) Necessity of training / expertise in security for stating software security requirements.

These issues can be seen in a positive angle by providing solutions as

1) Accommodation of new software security requirements as per the changing scenario of security related vulnerabilities;
2) Stating the software security requirements in a positive tone which make the validation easier compared to the negative tone statements;
3) Software security requirements must be language and platform independent;
4) A security requirement must be both testable and verifiable for it to be possible to track the progress of

the requirement throughout the phases, and test to ensure the requirement was included into the project.
5) A method can be established for ensuring the required software security requirements to be included as part of any application / software SRS (Software Requirement Specification) out of the listed software security requirements of GEOCHEMACS;
6) Fundamental awareness & training of secure programming and software security requirements engineering shall be provided to the development team and as well as to the management for inculcating the practices from security aspect.
It will be wise to take the major benefits of some of the well known methodologies. It has been quite essential to improve the software process and practicing with less ambiguity. That is why it has been selected a mix of multiple approaches such as CLASP (Comprehensive Lightweight Application Security Process) [2][8][55], Haley et al's Approach [2][5][7][21], Attack trees [8], misuse cases [8], SQUARE(Security Quality Requirements Engineering) [2][27] together with the nature of simplicity in practicing as the primary criteria. Primarily CLASP will ride the basic approach with the other four influencing and complimenting the practices in bringing software security requirements in GEOSCHEMACS. It is necessary to mention that various theses submitted and research done on software security requirements engineering are other inspirations behind selecting the mixed approach. The approach is basically security vulnerabilities based software requirements engineering with an internal perspective of the system.
While writing requirements it is necessary to make sure security is pervasive in all the key areas. The security requirements shall address the basics and also application vulnerabilities uncovered during threat modelling. SRS shall outline how security helps the overall problem being solved by the application, specific security goals, security controls inherent in the application's major features, security standards, security enabling detraction of each of use, security control limitations (that cannot be protected against), security features that enable user and system administration in security testing future application security testing. SRS (Software Requirement Specification) or some document shall make sure of the people such as developers, management, users are aware of security issues at hand.

III. SECURITY OBJECTIVES, POLICIES & PRACTICES

III.1. Security objectives

Security objectives for GEOSCHEMACS include building one higher abstraction level than software functional requirements for addressing system's security threats and enforcing organizational security policies. It shall address primarily the security issues with reference to building security while developing & enhancing GEOSCHEMACS software components by taking care of operational environment and related spacecraft considerations. Security objectives, if grouped based on the phases of treating security attacks can be preventive, detective and corrective. Preventive security objectives targeting on security mechanism that can effectively prevent attack before it has any effect on the system. Detective security objectives targets on security mechanisms that can quickly and effectively detect intrusion or abnormal behaviours if there is any, and covers subsequently actions or responses. Corrective security objective focuses on system recovery or data recovery when system or integrity of data is comprised.

III.1.1 Preventive security objectives

Identification & Authentication: Identifying all types of GEOSCHEMACS users using attributes such as separate login and passwords with reference to GEOSCHEMACS administration, spacecraft controlling, analysing and expert privileged accounts. *Confidentiality:* Protect information from unauthorized disclosure. Confidentiality is a complex security objective compared with others. It contains several aspects, or sub objectives: access control, information flow control, encryption, and residual data protection. Authorization is to make sure if user has the proper permission to perform actions (e.g., read, modify, delete) on a GEOSCHEMACS application and it is often required that the user is authenticated. While access control controls the access/operations on an SCHEMACS application based on some general rules, such as type of user, configured set of spacecrafts, type of system (server or client workstation or client command workstation) and personnel on shift for spacecraft control.
Integrity: Integrity protects information from unauthorized modification instead of disclosure. It is also possible that integrity is compromised by authorized users' mistakes. This is primarily of avoiding tampering intentionally or getting damaged

with reference to the mistakes caused by authorized users. The integrity checks at various levels such as application level, operation environment level and whole system level are necessary. Specific logging for specific applications for the actions by the users is required.
Availability: It ensures the accessibility of information and continuousness of system functionalities.
Non-repudiation: It ensures that the users of GEOSCHEMACS cannot deny that particular operation / action on any application has been done such as sending a prohibited command or doing an action which is temporarily put under "do not's". These are required to be taken care by logging specific events and some information at regular intervals.
Security Management: It provides users with certain roles the ability to customize the use of security mechanisms in GEOSCHEMACS such as enabling / disabling the security feature(s) of users or roles.
Privacy: Privacy targets on user's identity or actions non-observable to others.

III.1.2. Detective security objectives

These include accountability and intrusion detection & responding. Corrective security objective says about the recoverability.
Hence, it is necessary to support the security objectives of GEOSCHEMACS through properly mentioned software security requirements. Following are the security objectives which can be listed for the software security requirements of GEOSCHEMACS:
Ensuring of

- All types of users and GEOSCHEMACS applications are identified and their identities are properly verified.
- Authorised Users and GEOSCHEMACS applications can only access data and services
- Unathorised malicious programs or viruses do not infect the application or component or environment
- Practicing secure programming guidelines
- Detection of attempted intrusions by unauthorized persons and insider attackers and also unauthorized applications
- Not corrupting data and communication intentionally
- Non repudiation of the actions and interactions with the software & environment
- Confidential communication and data are kept private through encryption
- Auditing of the status and usage of security mechanisms
- Survival of attacks on software and operational environment, possibly running in degraded mode
- Not disturbing the security mechanism incorporated through GEOSCHEMACS applications during system maintenance
- Every application of GEOSCHEMACS is usable through a licensed key

III.2. Security policies, standards & practices

ISRO has a well defined policy for information security. Policies for software security shall be brought out as a separate entity to enhance & focus software security importance. Training and awareness programmes are required to be conducted across ISRO software community including developers, middle & top level management. As such the software life cycle international standard, IEEE 12207 has been practiced through ISPD (ISRO Software Process Document). This standard does not give any specific mention on software security requirements. This shall be enhanced with software security requirements as additional concern with reference to security vulnerabilities. The procedures and documents that have been produced and being produced shall be enhanced / incorporated for software security related issues. The policies, procedures and practices shall accommodate the software security concerns of complete control lying with any single individual.
A security control is a way to fulfill one or more security requirements. Security requirements fall into several categories, each of which can be satisfied by one or more security controls. The categories are
1) Permission to access data or exercise functionality
2) Verification of "who" and "what"
3) Securing information
4) Security policies, procedures and practices.

III.2.1 Guidelines

Security requirements shall not be specified in terms of the types of security architecture mechanisms that are typically used to implement them.
Ensure that few persons (who have been properly appointed on behalf of the organization that owns and controls the application or component) are able to authorize specific authenticated users and client applications to access specific application or component capabilities or information. Ensure that specific authenticated externals can access specific application or component capabilities or information if and only if they have been explicitly authorized to do so by a properly appointed person(s).

Authorization depends on both identification and authentication. Authorization shall be granted on the basis of user analysis and the associated operational requirements. Only a limited number of people (or roles) shall be appointed to grant or change authorizations.

GEOSCHEMACS can delegate immunity requirements to the spacecraft operation centre, but only if the centre provides (and will continue to provide) adequate security mechanisms to fulfil the requirements. This would be a legitimate architectural decision under certain circumstances.

Non repudiation requirements typically involve the storage of a significant amount of information about each interaction including the: Authenticated identity of all parties involved in the transaction, Date and time that the interaction was sent, received, and acknowledged (if relevant), Significant information that is passed during the interaction.

People and applications should have access only to the data and communications for which they are authorized.

Care should be taken to avoid unnecessary duplication between security-auditing and intrusion detection requirements. Survivability requirements are often critical for spacecraft control applications.

Physical protection requirements are related to survivability requirements. Survivability requirements specify continued functioning after an attack, whereas physical protection requirements specify the protection of components. Physical protection requirements are typically prerequisites for survivability requirements. System maintenance security requirements may conflict with operational availability requirements, in that the operational availability requirements may not allow one to take the application or component off-line during maintenance and the repetition of security testing.

III.2.2 Assessment and procedures

1. Identification and assessment of both of the logical targets of informational & processing resources and physical targets of hardware, LAN (Local Area Network) architecture shall be needed.
2. GEOSCHEMACS environment of corresponding hardware architecture & platform, version of installed firmware, operating system and it's version, installed software, enabled features, configuration parameters, peripherals & hardware specific software, interfaces shall be evaluated & looked with the perspective of security concerns and vulnerabilities.
3. Better information dissemination and response procedures for mitigating the vulnerability's impact shall be made available. Existence of risk analysis, incident response team and comprehensive & complete advisory description shall be made mandatory.
4. While assessing exploitation impact or damage, factors such as availability (denial of service), system or data integrity violation, loss of data, data disclosure & confidentiality breach, privilege elevation, stolen credentials, code/script execution, bypass of intended controls, misuse of resources, violation of system's security policy, affecting neighbour systems (spreading), erroneous transmission and physical damage shall be considered.
5. Financial loss (labour time loss), loss of trust, personal abuse, defamation & humiliation, unauthorised gain of political authority and status, blackmail and other criminal action, action against the law, effects on national security and defense shall be assessed for the incident of damage.
6. Procedures for the implementation of solutions such as patching & configuring according to the relevant security advisories shall be made available. Additional protection measures might be required such as ACLs (Access Control Lists), intrusion detection systems, firewalls, cryptography, virtual private networks and antivirus applications. Collection of reliable evidence data by means different loggings is mandatory.

IV. ASSET IDENTIFICATION

An adversary will not attack a system unless there is an asset available, so identifying assets is key in determining how the system needs to be protected. Asset identification can be done through different viewpoints: the customer's, the system owner's, and the attacker's. After identifying the assets, further analysis should be done on each to identify a priority on protection. This will ensure that the most valuable assets receive the most attention in threat mitigation.

Assets include data, software & hardware components and communication services. The assets that can be listed for GEOSCHEMACS are as follows:

IV.1. Data and software

1. Spacecraft raw telemetry, telecommand and tracking data from spacecraft control centre and remote stations.
2. Spacecraft health monitoring and controlling & validating related telemetry parameter and telecommand data base.
3. GEOSCHEMACS administration and configuration data files.
4. Spacecraft ground control required telecommand

code data files.
5. Spacecraft daily operation schedule files.
6. Spacecraft ground auto controlling required event files.
7. Contingency recovery related files.
8. GEOSCHEMACS client and server configuring related files for the required list of spacecrafts.
9. GEOSCHEMACS executables.
10. User account(s) profile files.
11. Processed statistics data.
12. Operator input files used for GEOSCHEMACS analysis software.
13. Spacecraft specific special payload data.
14. Spacecraft monitoring required normal and critical alarm limit files.
15. GEOSCHEMACS source code.

IV.2 Hardware components

1. TM (Telemetry) access devices.
2. Telecommand encoding devices.
3. Tracking data access devices.
4. GEOSCHEMACS offline data servers.
5. GEOSCHEMACS acquisition and real-time data servers.
6. GEOSCHEMACS client workstations.
7. Payload data acquisition and distribution servers.
8. WEBGEOSCHEMACS servers.
9. Communication devices such as routers, switches and multiplexers.
10. Auxiliary stored / history data media drive such as digital tape drive, CD/DVD drive and Floppy drive and USB port access.

IV.3. Users

1. SCHEMACS administrators and software managers.
2. Spacecraft controller/operators.
3. Shift controller/manager.
4. Spacecraft analyzers and spacecraft experts.
5. Higher authorities.
6. Spacecraft data base managers.
7. Users at spacecraft design and manufacturing centre.
8. SCHEMACS designers.
9. Spacecraft subsystem experts.

Prioritising among these is bit difficult but with reference to the space services supporting & business goals, spacecraft commanding is considered as the utmost important task. The assets of software, hardware and data related to commanding shall be given highest priority. These may include TM access devices, Telecommand encoders, GEOSCHEMACS acquisition and real-time data & commanding servers, GEOSCHEMACS real-time data acquisition, processing and commanding software components, real-time presentation software, Spacecraft health monitoring and controlling & validating related telemetry parameter and telecommand data base, Spacecraft ground control required telecommand code data files, GEOSCHEMACS administration and configuration data files, Spacecraft controller/operators, communication devices such as switches. The next and middle level priority can be allocated to assets of GEOSCHEMACS offline data servers, GEOSCHEMACS client workstations, Tracking data access devices, software components related to TM (Telemetry), Telecommand and tracking data archival, tracking data acquisition & processing software elements, client workstation based software components, offline analysis software package, ground operations & defined spacecraft characteristics auto monitoring & reaction automation software, Spacecraft ground auto controlling required event files, Spacecraft daily operation schedule files, Contingency recovery related files, Spacecraft monitoring required normal and critical alarm limit files, Operator input files used for GEOSCHEMACS analysis software, SCHEMACS administrators and software managers, Spacecraft analyzers and spacecraft experts. Another priority of next importance can be allotted to the remaining assets but for few exceptions such as experimental payload data & related which can be given the lowest priority.

V. THREATS, ATTACK TREES & MISUSE CASES

Security engineering is about building systems that are and can remain dependable in the face of malice, error or mischance. As a discipline, security engineering focuses on the tools, processes and methods needed to design, implement and test complete systems to adapt existing systems as their environment evolves. The goal of software security engineering is to build better, defect free software. Software intensive systems that are constructed using more securely developed software are better able to continue operating correctly in the presence of most attacks by either resisting the exploitation of weakness in the software by attackers or tolerating the failures that result from such exploits. A security environment describes the context in which the software is expected to evolve. The environment affects the kind of threats the application is likely to encounter. Threat is a potential for harm of an asset. Attack is an action intended to violate the security of an asset. Attacker is an entity that carries out attacks. Risk is the probability that a successful attack occurs. The most listed threats include theft, vandalism, unauthorized disclosure, destruction, fraud, extortion, espionage, trespass. Attackers include crackers, disgruntled employees, cyber terrorists, spies and even novice idle experimenters. Among these who could be adversaries? What types of attacks or misuse cases are possible? What could be the ways of protecting? These questions are to be answered before documenting the security requirements.

It is required to find out misuse cases. Threats are identified and found based on the experiences and past projects. Threat modeling is an attempt to capture the thought process of an adversary that wants to achieve a set of goals on the system. Threat modeling [2][35] takes the viewpoint of a potential adversary interacting with the system. Any methods of interaction with the system are potential entry points. A threat does not exist unless an entry point leads to access of an asset. Any attack that is not mitigated or is mitigated improperly has a vulnerability that could be exploited to gain access to the asset that the system protects. The most common diagram for enumerating threats is attack tree. Like use cases are for requirements, misuse cases can be used for forming security requirements. Threat model describes the possible threats that can occur in a given security environment. Attack tree can be built to detail each threat with the attack's goal represented as the tree's root and leaf nodes representing different ways to achieve that goal. A general solution shall be brought out to counter each threat. A document shall be maintained for recording threats. Security solutions shall be documented along with the threat & misuse cases. The solution shall be described with the flow of events with reference to system, users and attacker activities. With ranking or priority or risk associated for each threat, the evaluation of risk for the threat can be analysed. Elimination of threats at an acceptable risk level and with trust assumptions [21], the list of threats to be mitigated can be identified.

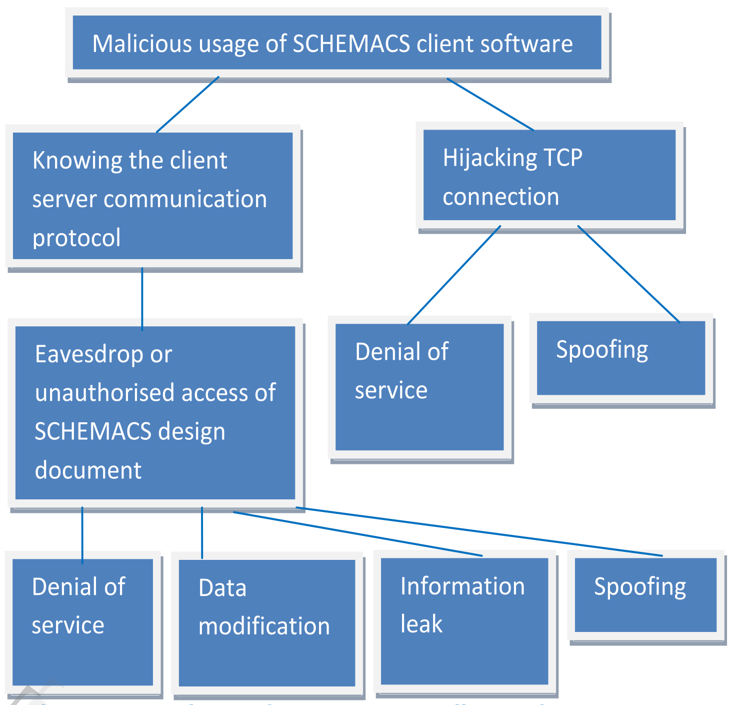
Malicious usage of SCHEMACS client software
Knowing the client server communication protocol
Hijacking TCP connection
Eavesdrop or unauthorised access of SCHEMACS design document
Denial of service
Spoofing
Denial of service
Data modification
Information leak
Spoofing

Figure 3. Attack Tree for SCHEMACS client software malicious usage

Various software security vulnerabilities include buffer overflows, stack overflows, heap overflows, invalidated input, race conditions (time of check–time of use and inter process communication), insecure file Operations, access control problems, secure storage and encryption. Buffer overflows occur when a given size X is allocated in memory but more than X is written. The additional bytes overwrite the memory, a bug that lets an attacker write in memory he or she should not have access to and that can be exploited in various malicious ways. For better understanding system vulnerabilities, proactive security services, teams [58] are used for simulating the attacks. These teams help in finding software vulnerabilities and find ways & practices, countermeasures for mitigating / defending attacks from script kiddies to well motivated hackers. An insider threat model can be used adversary simulation describing a malicious insider using various attributes such as access, knowledge, privileges, skills, risk, tactics, motivation and process. This suits for GEOSCHEMACS. Another threat model [35] developed by Michael Howard and David Leblanc which uses an iterative approach to assess the vulnerabilities in a given application can also be used for GEOSCHEMACS software. This model begins with a functional decomposition of an application using data flow diagrams emphasizing that the more is known about an application, the easier it is to uncover the threat targets hidden in the application.

Threat descriptions can be represented as tuples of the form {threat, asset, damage / impact}. Using the identified list of assets, violation on the general security goals such as confidentiality, integrity and availability and the corresponding damage or harm can be listed through threat descriptions for using in elicitation of security requirements. These threat descriptions are traced against one or more security requirements or one security requirements can mitigate one or more threats.
Since GEOSCHEMACS is operated in an isolated mission control network, GEOSCHEMACS software can be added better security requirements using a combination of the methods of insider threat model, a threat model using [35] DFDs (Data Flow Diagram) of the applications, attack trees, threat descriptions and misuse cases.

VI. ELICITATION OF SOFTWARE SECURITY REQUIREMENTS

The approach of positive tone stating of software security requirements and non dependency on any language & platform is the primary line in eliciting software security requirements for GEOSCHEMACS. Software security requirements can be defined as the constraints on the functional requirements of the system, where these constraints operationalise one or more security objectives and goals. Security requirements like functional requirements are prescriptive in providing a specification to achieve the desired effect. Most of the successful attacks on software result from successful targeting and exploitation of known but non-practical vulnerabilities or unintentional misconfiguration. That is why primary focus on security vulnerabilities related security requirements shall be given.

1. The threat descriptions and attack trees along with mitigation techniques and solutions for not violating security goals & objectives shall be brought out as threat specification document.

VI.1. Software requirements related to vulnerabilities

Following 2 to 33 numbered requirements are the excerpts from the thesis [2].

2. When copying data into a buffer, the application shall ensure that the data being copied does not exceed the bounds of the buffer. If data being copied into a buffer is expected to have a termination character to determine end of the data, then the application shall ensure that the termination character is present in the data, and ensure that the data being copied does not exceed the bounds of the buffer.
3. The application shall not contain code that either directly or indirectly causes the instruction pointer to load with an address outside of instruction space.
4. The application shall ensure the value of an environment variable is in expected format before use. If an environment variable used by the application is expected to have a set of values, the application shall ensure the value of the environment variable is one of those expected values before use. Before invoking another application for execution, the invoking application shall clear all environment variables, and set environment variables required for execution with trusted values for the application being invoked.
5. After an application makes a request for memory, the application shall check to see that the memory was properly allocated, and only use memory that has been successfully allocated.
6. The application shall store sensitive data in memory only when necessary, and use a subroutine provided by the operating system to store the sensitive data in an encrypted format when the sensitive data is not in active use. The application shall not reallocate memory containing sensitive data.
7. If an application holds sensitive data in memory, the application shall write zeros to the entire block of memory before releasing the memory to the operating system. The application shall not use a built in subroutine for writing the zeros.
8. After a pointer to a memory location on the heap is deallocated, the application shall set that pointer's value to NULL. Before dereferencing a pointer, the application shall ensure that the pointer's value is not set to NULL. The application shall only set a pointer's value to an address located on the stack, if the lifetime of the stack variable will end after the lifetime of the pointer.
9. The application shall never make a request to the operating system for zero bytes of memory.
10. The application shall ensure that all addresses placed in a pointer variable are to memory locations other than its own.
11. The application shall ensure the value of string input is in expected format before use. If an input has an expected set of values, the application shall ensure the value of the input is one of those expected values before use.
12. Before an addition operation where both operands are positive, the application shall ensure that subtracting the left operand from the largest

possible value is greater than or equal to the right operand. When the previous condition is not met, the application shall not perform the addition operation, and block any corresponding input from further use. Before an addition operation where both operands are negative, the application shall ensure that subtracting the right operand from the smallest possible value is less than or equal to the left operand. When the previous condition is not met, the application shall not perform the addition operation, and block any corresponding input from further use.

13. Before a subtraction operation with unsigned numbers, the application shall ensure that the left operand is greater than the right operand. When the previous condition is not met, the application shall not perform the subtraction operation, and block any corresponding input from further use. Before a subtraction operation where the left operand is nonnegative and the right operand is negative, the application shall ensure that adding the right operand to the largest possible value is greater than or equal to the left operand. When the previous condition is not met, the application shall not perform the subtraction operation, and block any corresponding input from further use. Before a subtraction operation where the left operand is negative and the right operand is positive, the application shall ensure that adding the right operand to the smallest possible value is less than or equal to the left operand. When the previous condition is not met, the application shall not perform the subtraction operation, and block any corresponding input from further use.
14. Before a multiplication operation where both operands have the same sign, the application shall ensure that dividing the right operand by the largest possible value is greater than or equal to the left operand. When the previous condition is not met, the application shall not perform the multiplication operation, and block any corresponding input from further use. Before a multiplication operation where operands have different signs, the application shall ensure that dividing the right operand by the smallest possible value is less than or equal to the left operand. When the previous condition is not met, the application shall not perform the multiplication operation, and block any corresponding input from further use.
15. The application shall ensure that a division operation never contains the largest negative value in the numerator, with a -1 in the denominator. When the previous condition is not met, the application shall not perform the division operation, and block any corresponding input from use.
16. Before assigning a new value to a variable used as an index to a buffer, the application shall ensure the new value is within the buffers bounds.
17. Before assigning a new value to a variable used to hold the length or quantity of an object, the application shall ensure that the new value is nonnegative.
18. The application shall not store sensitive data in static memory.
19. When creating a new file/directory, the application shall set the access permissions so the fewest number of users possible have access. The application shall set the access permissions using the file descriptor and not the filename.
20. Before using a file/directory, the application shall check the ownership and the access permissions of the file/directory, and only use files/directories with expected ownership and access permissions.
21. When creating a file, the application shall ensure the path and name are unique.
22. If the application does not need to open files/directories through links, the application shall use a file open subroutine that blocks the opening of files/directories through links. If the application needs to open files/directories through links, the application shall first check the access permissions of the link itself. Then, the application shall open the file, and use the file descriptor to check the access permissions of the file. If the access permissions of the file and link do not match, the application shall not use the file.
23. The application shall ensure that data contained in a file is in expected format before use. If a file used by the application is expected to have a set of data, the application shall ensure that the expected data is in that file before use.
24. After opening a file, but before reading or writing to that file, the application shall request the operating system lock the file, using the file descriptor returned by the file open operation. Before closing a file, but after all reading and writing operations have been completed on that file, the application shall request the operating system unlock the file, using the file descriptor given to the previous lock file operation.
25. If writing sensitive data to a file, the application shall encrypt and provide authentication for the data. The application shall provide authentication for unencrypted data in files. The application shall only use data from files that have verified authenticity.

26. After deleting a file, if the data is too sensitive to be left on disk even in an encrypted format, the application shall write zeros to the entire hard-drive partition that contained the file. The application shall take into consideration that the procedure could also remove the operating system from disk.
27. After attempting a write to a file, the application shall ensure that the write was successful.
28. If writing sensitive data to a file, the application shall encrypt and provide authentication for the data. The application shall provide authentication for unencrypted data in files. The application shall only use data from files that have verified authenticity.
29. When receiving encrypted data over a network, the application shall decrypt and authenticate all data before its use, and reject data that does not authenticate or decrypt properly. When receiving unencrypted data over a network from another host, the application shall authenticate all data before its use, and reject data that does not authenticate properly.
30. The application shall authenticate the remote host to verify its identity before sending or accepting any data from that host. The application shall ensure data received from a remote host is in expected format before use. The application shall only send data in the expected format.
31. If an application is sending sensitive data to another host on a network, the application shall encrypt and provide authentication for the data. When sending data over a network to another host, the application shall provide authentication for the data.
32. The application shall check to see if the desired TCP port is available for listening before waiting for connections on that port. The application shall check to see if a connection to a remote host was accepted before sending data to that host.
33. The application shall send all numerical data over a network in network byte order. The application shall expect all numerical data received over a network to be in network byte order.
34. The application shall handle the errors properly for all the system routines and library function calls by applying fail safe method.
35. Inter process communication methods (such as shared memory, message queues, semaphore sets etc..) used in GEOSCHEMACS shall be created and maintained with appropriate and minimum required access permissions for posting & retrieving so that unauthorized access can be thwarted.
36. The application shall avoid sharing signal handler routine and the signal handler routines shall be simple.
37. The application shall use reentrant safe function calls in signal handler routines.
38. The application shall create child process securely such that no extra privileges are passed to the child.
39. The application with setuid or setgid shall drop the privileges once after the required work with privileges is over.
40. The application shall limit the number of sockets being created, the number of threads being spawned.
41. The application shall limit the allocation of buffers on the stack and the number of file system reads & writes.
42. The application shall avoid writing configuration files to world accessible directories.
43. The application shall use copy functions that copy a maximum number of bytes rather than ones that rely on NULL terminated strings only.
44. The application shall coin unique names for temporary files and shall ensure the temporary files are protected from removing before the usage & it's lifetime.
45. The application shall use file handling functions that identify files using file descriptors.
46. The application shall avoid filenames with leading dashes, with control characters and with spaces.
47. The application shall check the standard input, standard output and standard error file descriptors are open; if not, it shall open them using /dev/null.
48. Any application shall avoid opening a new file with a fixed file descriptor.
49. The application shall ensure to provide a format string argument.

Requirements related to cryptography have not been included since utilization & necessity of cryptographic resources are in specific software only and also have been categorized under confidential.

VI.2. Identification requirements [1]

50. The application shall identify all of its client applications before allowing them to use it's capabilities. The application shall identify all of its human users before allowing them to use capabilities.
51. GEOSCHEMACS using centre shall identify all personnel before allowing them to enter to specific

places such as server rooms, mission control centre, etc.

52. Any application shall have single sign-on during a single session of usage. If required each application may ensure the name of the user against payroll database or employee database.

VI.3 Authentication requirements [1]

The authentication requirements for GEOSCHEMACS can be:

53. The application shall verify the identity of all of its users before allowing them to use its capabilities. The application shall verify the identity of all of its users before allowing them to update their user information.
54. GEOSCHEMACS using centre shall verify the identity of all personnel before permitting them to enter to specific places such as server room, mission control centre etc.
55. GEOSCHEMACS applications where two or more levels of authentication required shall be ensured for the login attempts to the corresponding applications are restricted to an identified number with reference to the current working terminal.

VI.4 Authorisation requirements

56. Any application in GEOSCHEMACS shall allow each user to obtain access to all of user's account information only with the necessary privileges and permissions granted [1].
57. Every application under GEOSCHEMACS shall implement the license key for it's operation so that unauthorized execution and access can be controlled.

VI.5 Immunity requirements

58. The application shall protect itself from infection / attack by ensuring the capabilities of validating all entered data [1]. The application shall notify the user / administrator if any violation of input data. The application shall get developed using secure programming practices.
59. GEOSCHEMACS operational account related login profiles shall be protected for modifications and deletions.
60. Every application shall undergo an identified (or in house built) and customized code review tool for secure programming coding standards & practices.
61. GEOSCHEMACS operational environment shall be seen with disabling of memory crash dumps of the applications.

VI.6 Integrity requirements [1]

62. GEOSCHEMACS application shall validate and protect the corruption of data for the unintentional mistakes or data access across multiple spacecrafts and also protect the corruption / deletion / modification of any TM or transmitted Telecommand data coming from external users, remote stations and communication services. Good secure programming practices shall be implemented into the applications.

VI.7 Intrusion detection requirements [1]

63. The application shall detect and record all attempted accesses that fail identification, authentication, or authorization requirements.

VI.8 Non repudiation requirements [1]

64. GEOSCHEMACS software shall record all important events and log every Telecommand transmission. The application shall log every interaction of the user for mission database editing during the current session.
65. The application shall log or record the modification / addition / deletion of any configuration or configuration data file.

VI.9 Privacy requirements [1]

66. GEOSCHEMACS shall ensure of the privacy of sensitive data and shall ensure of not having any personal information kept in part of data or communication. The application shall make the required data only available for the authorised people for the requests put on.
67. GEOSCHEMACS software shall ensure to apply isolation of data and applications access from mission control network to any remote stations and other public networks.

VI.10 Security auditing requirements [1]

68. GEOSCHEMACS shall record, summarise and report the status of security mechanisms involved and also the incidents of attacks or breaches to appropriate authority or security auditing team.

VI.11 Survivability requirements [1]

69. GEOSCHEMACS applications shall be developed for avoiding single point of failure. The applications shall be able to continue to work possible extent (mostly in degraded mode) even in the case of any attack, destruction of data or environment.

VI.12 Physical protection requirements [1]

70. GEOSCHEMACS operational environment shall ensure the protection of hardware, software, data against physical damage, theft, replacement, destruction and sabotage.

VI.13 System maintenance requirements [1]

71. GEOSCHEMACS software shall ensure all types of security requirements are complied during any enhancement or upgrading of data, hardware or software component. The application shall ensure the integrity of data, environment with reference to the enhancement and/or maintenance.

VI.14 Web application based security requirements

The software security requirements specific & additional for WEBGEOSCHEMACS applications can be stated as follows:

72. Every web application shall validate the input properly and preferably at server level to avoid buffer overflows, SQL injection and cross site scripting.
73. Web applications shall log important events with sufficient data for enabling the administrator to detect the attacks, errors, non repudiation issues and for recovery from the attacks.
74. Web applications shall incorporate more than one defense countermeasures to discourage potential attackers.
75. Web applications shall enable the minimum and required privilege for data & functionality access.
76. Web applications shall validate all URIs / URLs.
77. Web application shall forbid HTTP GET to perform non queries.

VI.15 Prioritisation, compromising and selection

The selection of the necessary software security requirements from the list elicited above into specific SRS can be based on the severity of impact of the attack on the software & related asset(s), the trust assumptions and the level of compromising with the corresponding requirements. It is impossible to produce cent percent secure software and environment for GEOSCHEMACS. The inclusion of all types of security requirements into the SRS and incorporating in the software during the development phase will certainly give high quality and secure software & environment and inturn saves huge amount of time & money. Specifying security features at the SRS ensures that acceptance tests include testing for security features, a measure which significantly improves the security assurance of the software being produced. Prioritisation of any misuse cases and incorporation of the protection against them shall be as per the identified asset priority and severity of impact. Trust assumptions are the basic guidelines that are assumed to be made available or followed. Some of the requirements under software security can be compromised with the procedure & policy guidelines assured and practiced. The requirements which can be under guidelines, procedures and trust assumptions can be deliberated with user & developer community and as per the type & priority of software.

VI.16 Verification & validation

Since all the software requirements have been stated in positive tone and are the constraints on functionality, each of the software security requirements can be verified in the same manner what is practiced for functional requirements. Security requirements typically require security specific testing in addition to the traditional types of testing. Test cases may be based on misuse cases; load & stress testing may be useful in validating. Security requirements shall be validated for satisfying the security goals. Arguments shall be produced for validating trust assumptions.

VII. CONCLUSION

Addition of security concern and incorporation of the security objectives right from the initial phase of requirements is the beginning of the path towards secure development of GEOSCHEMACS. The next phases of life cycle such as design, coding and system testing shall also be continued with specific additions & practices into the development process of GEOSCHEMACS. Secure programming awareness & training shall be another factor for enhancing the software process towards secured assurance. The software security requirements mentioned above are implementable into next phases; they are concise, testable and also verifiable. The above requirements are specific to software vulnerabilities which are the root cause of exploits. This software security requirements engineering would serve as an initial step for GEOSCHEMACS programming securely.

REFERENCES

[1] Engineering Security Requirements: Donald G Firesmith, Firesmith Consulting, USA; Journal of Object Technology, Vol 2, No 1, January-February 2003.
[2] Thesis submitted on "Security Requirements for the Prevention of Modern Software Vulnerabilities and a Process for Incorporation into Classic Software Development Lifecycles" by Lee M. Clagett II.
[3] http://csrc.nist.gov/cc/ccv20/ccv2list.html
[4] A thesis submitted on "Security Functional Requirements Analysis for developing secure software" by Dan Wu
[5] "Security Requirements Engineering: A Framework for Representation and Analysis" by Charles B. Haley, Robin Laney, Jonathan D. Moffett, Member, IEEE, and Bashar Nuseibeh, Member, IEEE Computer Society; IEEE transactions on Software Engineering, Vol. 34, No. 1, January/February 2008.
[6] "Security Requirements Engineering for Software Systems: Case Studies in Support of Software Engineering Education" by Nancy R. Mead and Eric D. Hough; Proceedings of the 19th Conference on Software Engineering Education & Training (CSEET'06).
[7] A thesis submitted on "Arguing Security: A Framework for Analyzing Security Requirements" by Charles B. Haley.
[8] "Security Requirements Engineering: A Survey" by Jose Romero-Mariona, Hadar Ziv, Debra J. Richardson; University of California, Irvine.
[9] "Security Requirements Engineering Framework for Software Product Lines" by Daniel Mellado, Eduardo Fernández-Medina2 and Mario Piattini.
[10] "Security Requirements Engineering: When Anti-requirements Hit the Fan" by Robert Crook*, Darrel Ince, Luncheng Lin, Bashar Nuseibeh.
[11] "Towards a Risk-Based Security Requirements Engineering Framework" by Nicolas Mayer, Andr´e Rifaut, Eric Dubois.
[12] "A Meta-Model for Usable Secure Requirements Engineering" by Shamal Faily and Ivan Fléchais.
[13] "Introduction to Software Engineering for Secure Systems" by Danilo Bruschi, Bart De Win and Mattia Monga.
[14] "Software Security Requirements Gathering Instrument" by Smriti Jain and Maya Ingle; *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 7, 2011*.
[15] "Software Security Rules: SDLC Perspective" by C. Banerjee, S. K. Pandey; (IJCSIS) International Journal of Computer Science and Information Security, Vol. 6, No.1, 2009.
[16] "Security Requirements Engineering – A Strategic Approach" by Dr Alagarsamy K and Chandrabose A.
[17] "Security-aware Software Development Life Cycle (SaSDLC) – Processes and Tools" by Asoke K Talukder, Vineet Kumar Maurya, Santhosh Babu G, Jangam Ebenezer, Muni Sekhar V, Jevitha K P, Saurabh Samanta, Alwyn Roshan Pais.
[18] "Information Security Anti patterns in Software Requirements Engineering" by Miroslav Kis
[19] "Incorporating Security Requirements from Legal Regulations into UMLsec model" by Shareeful Islam P, Jan Jürjens.
[20] "Security Requirements Engineering: State of the Art and Practice and Challenges" by Golnaz Elahi.
[21] "Using Trust Assumptions in Security Requirements Engineering" by Charles B. Haley, Robin C. Laney, Bashar Nuseibeh, Jonathan D. Moffett.
[22] "Security and Trust Requirements Engineering" by Paolo Giorgini, Fabio Massacci, and Nicola Zannone.
[23] "A Novel Model for Security Requirements Process" by Hui Wang, Shulan Gao, Bibo Lu, Zihao Shen.
[24] "Extending XP Practices to Support Security Requirements Engineering" by Gustav Boström, Jaana Wäyrynen and Marine Bodén.
[25] "Review of Security Requirements Engineering Processes" by Brendan Cervin.
[26] "Requirement Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard" by Paolo Giorgini and Fabio Massacci1 John Mylopoulos.
[27] "Security Quality Requirements Engineering (SQUARE) Methodology" by Nancy R. Mead and Ted Stehney.
[28] "Security Requirements Engineering; State of the Art and Research Challenges" by M. A. Hadavi, V. S. Hamishagi, H. M. Sangchi.
[29] "Agile Security Requirements Engineering" by Johan Peeters
[30] "How to capture, model, and verify the knowledge of legal, security and privacy experts: a pattern-based approach" by Luca Compagna, Paul El Khoury and Fabio Massacci.
[31] Thesis submitted on "Capturing Software Systems Security Requirements" by Hassan EL-Hadary.
[32] "Capturing Security Requirements through Misuse Cases" by Guttorm Sindre Andreas L. Opdahl.
[33] "Goal-Oriented Security Trade-Off Modeling and Analysis with Knowledge Support" by Golnaz Elahi
[34] "THREAT-DRIVEN ARCHITECTURAL DESIGN OF SECURE INFORMATION SYSTEMS" by Joshua Pauli.
[35] "Threat Modeling as a Basis for Security Requirements" by Suvda Myagmar, Adam J. Lee and William Yurcik.
[36] "Using Security Patterns to Model and Analyze Security Requirements" by Betty H.C. Chengy, Sascha Konrad, Laura A. Campbell, and Ronald Wassermann.
[37] "RUPSec: An Extension on RUP for Developing Secure Systems - Requirements Discipline" by Mohammad

Reza Ayatollahzadeh Shirazi, Pooya Jaferian, Golnaz Elahi, Hamid Baghi, Babak Sadeghian.

[38] "A MODEL-DRIVEN APPROACH TO ARCHITECTING SECURE SOFTWARE" by Ebenezer A. Oladimeji, Sam Supakkul Lawrence Chung

[39] "Trade-off Analysis of Misuse Case-based Secure Software Architectures: A Case Study" by Joshua J. Pauli, Dianxiang Xu

[40] "Cutting Edge Practices for Secure Software Engineering" by Kanchan Hans

[41] "SECURITY THREAT MODELING AND ANALYSIS: A GOALORIENTED APPROACH" by Ebenezer A. Oladimeji, Sam Supakkul and Lawrence Chung

[42] "Effective Security Impact Analysis with Patterns for Software Enhancement" by Takao Okubo, Haruhiko Kaiya and Nobukazu Yoshioka

[43] "Security Requirements Analysis, Specification, Prioritization and Policy Development in Cyber-Physical Systems" by Kenneth Kofi Fletcher and Xiaoqing (Frank) Liu

[44] "Security Requirements Addressing Security Risks for improving Software Quality" by Shareeful Islam, Wei Dong

[45] "Hierarchy-Driven Approach for Attack Patterns in Software Security Education" by Joshua J. Pauli, Patrick H. Engebretson

[46] "Misuse Case-Based Design and Analysis of Secure Software Architecture" by Joshua J. Pauli and Dianxiang Xu

[47] "Effective Security Requirements Analysis:HAZOP and Use Cases" by Thitima Srivatanakul_, John A. Clark, and Fiona Polack

[48] ***"Suraksha:*** A Security Designers' Workbench" by Vineet Kumar Maurya, Santhosh Babu G, Jangam Ebenezer, Muni Sekhar V, Asoke K Talukder, Alwyn Roshan Pais

[49] "A Security Requirements Approach for Web Systems" by Stefan Wagner, Daniel Mendez Fernandez, Shareeful Islam, and Klaus Lochmann

[50] http://www.cert.org

[51] http://www.CommonCriteriaportal.org/public/expert/

[52] http://www.schneier.com/paper-attacktrees-ddj-ft.html

[53] http://searchsoftwarequality.techtarget.com/tip/0289483sid92-gci118682100.html

[54] http://sites.google.com/site/ijcsis/

[55] Building security requirements with CLASP by John Viega

[56] Secure software development by example; by Axelle Apvrille and Makan Pourtandij; IEEE security & privacy July / August 2005.

[57] Practical Internet Security By Vacca John R, Springer.

[58] Towards an automated attack model for red teams; by Helayne T rAy, Raghunath vemuri, Hariprasad R Kantubhukta; IEEE security & privacy July/August 2005.

[59] Applied Cryptography, Bruce Schneier, 2nd edition, WSE wiley.

[60] Cryptography and Network security, William stallings, Prentice Hall.

[61] Secure Programming Cookbook for C and C++ by Matt Messier and John Viega.

[62] Secure Programming for Linux and Unix HOW TO by David A.Wheeler.