decision. As part of the procurement methodology and process, in addition to the functional software requirements, secure software requirements must also be communicated and appropriately evaluated. Additionally it is important to include software security requirements in legal protection mechanisms such as contracts and SLAs. The need for software escrow is an important requirement when procuring software. The Software Acceptance chapter and the Supply Chain Security chapter will cover these concepts in more detail.

Protection Needs Elicitation (PNE)

In addition to knowing the sources for security requirements and the various types of secure software requirements that need to be determined, it is also important to know the process of eliciting security requirements. The determination of security requirements is also known as protection needs elicitation (PNE). PNE is one of the most crucial processes in information systems security engineering. For PNE activities to be effective and accurate, strong communication and collaboration with stakeholders is required, especially if the stakeholders are non-technical business folks and end users. With varying degrees of importance placed on security requirements, combined with individual perceptions and perspectives on the software development project, PNE activities have been observed to be a challenge.

PNE begins with the discovery of assets that need to be protected from unauthorized access and users. The Information Assurance Technical Framework (IATF) issued by the United States National Security Agency (NSA) is a set of security guidelines that covers Information Systems Security Engineering (ISSE). It defines a methodology for incorporation assurance/security requirements for both the hardware and software components of the system. The first step in the IATF process is PNE which is suggested to be conducted in the following order:

- Engage the customer
- Information management modeling
- Identify least privilege applications
- Conduct threat modeling and analysis
- Prioritize based on customer needs
- Develop information protection policy
- Seek customer acceptance

PNE activities may be conducted in several ways as *Figure 2.9* illustrates.

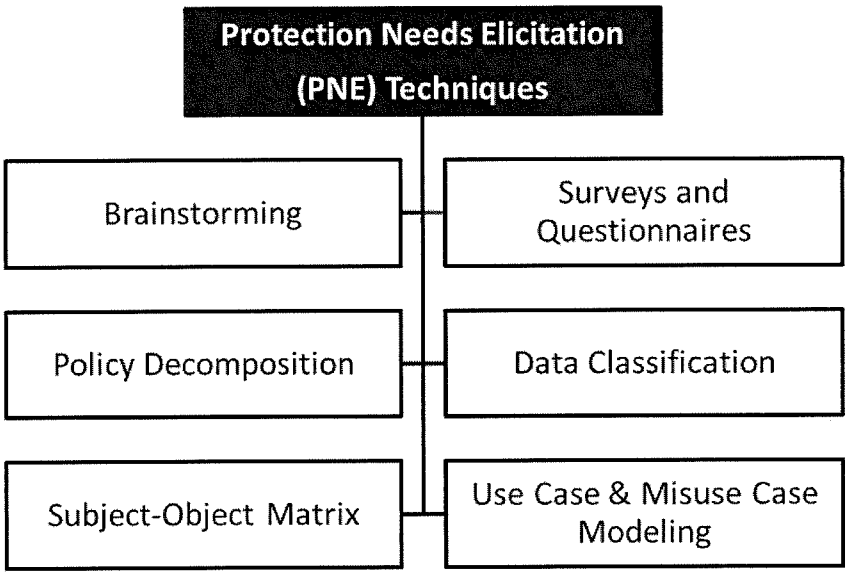

Figure 2.9 – **Protection Needs Elicitation (PNE) Techniques**

Some of the most common techniques to elicit protection needs (security requirements) include:

- Brainstorming
- Surveys (Questionnaires and Interviews)
- Policy Decomposition
- Data Classification
- Subject-Object Matrix
- Use Case & Misuse Case Modeling

Brainstorming

Brainstorming is the quickest and most unstructured method to glean security requirements. In this process, none of the expressed ideas on security requirements are challenged but instead they are recorded. While this may allow for a quick-and-dirty way to determine protection needs, especially in rapid application development situations, it is not advised for PNE because it has several shortcomings. First there is a high degree of likelihood that the brainstormed ideas don't directly relate to the business, technical and security context of the software. This can either lead to ignoring certain critical security considerations or going overboard on a non-trivial security aspect of the software. Additionally, brainstorming solutions are usually not comprehensive and consistent because it is very subjective. Brainstorming may be acceptable to

determine preliminary security requirements but it is imperative to have a more structured and systematic methodology for consistency and comprehensiveness of security requirements.

Surveys (Questionnaires and Interviews)

Surveys are effective means to collect functional and assurance requirements. The effectiveness of the survey is dependent on how applicable the questions in the surveys are to the audience that is being surveyed. This means that the questionnaires are not a one size fits all type of survey. This also means that both explicitly specified questions as well as open ended questions should be part of the questionnaire. The benefit of including open ended questions is that the responses to such questions can yield security related information which may be missed if the questions are very specific. Questionnaires developed should take into account *business* risks, *process (or project)* risks and *technology (or product)* risks. It is advisable to have the questions developed so that they cover elements of the software security profile and secure design principles. This way, the answers to these questions can be directly used to generate the security requirements. Some examples of questions that be asked are:

- What kind of data will be processed, transmitted or stored by the software?
- Is the data highly sensitive or confidential in nature?
- Will the software handle personally identifiable information or privacy related information?
- Who are all the users who will be allowed to make alterations and will they need to be audited and monitored?
- What is the maximum tolerable downtime for the software?
- How quickly should the software be able to recover and restore to normal operations when disrupted?
- Is there a need for single sign-on authentication?
- What are the roles of users that need to be established and what privileges and rights (such as create, read, update or delete) will each role have?
- What are the set of error messages and conditions that you would need the software to handle when an error occurs?

These questions can be either delivered in advanced using electronic means or asked as part of an interview with the stakeholders. As a CSSLP, it is expected that one will be able to facilitate this interview process. It is also a recommended

practice to include and specify a scribe who records the responses provided by the interviewee. Like questionnaires, the interview should also be conducted in an independent and objective manner with different types of personnel. Additional PNE activities may be necessary, especially if the responses from the interview have led to new questions that warrant answers. Collaboration and communications between the responders and the interviewers are both extremely important when conducting a survey based security requirements exercise.

Policy Decomposition

One of the sources for security requirements is internal organizational policies that the organization need to comply with. Since these policies contain in them high level mandates, they need to be broken down (or in other words decomposed) into detailed security requirements. However, this process of breaking high level mandates into concrete security requirements is not limited only to organizational policies. External regulations, privacy and compliance mandates can also be broken down to glean detailed security requirements. To avoid any confusion, for the remainder of this chapter, we will refer all these high level sources of security requirements as *policy documents*, regardless of whether they are internal or external in their origin.

While superficially it may seem as the policy decomposition process may be pretty simple and straightforward, since policies are high level and open to interpretation, careful attention is paid to the scope of the policy. This is to ensure that the decomposition process is objective and compliant with the security policy, and not merely someone's opinion. The policy decomposition process is a sequential and structured process as illustrated in *Figure 2.10*.

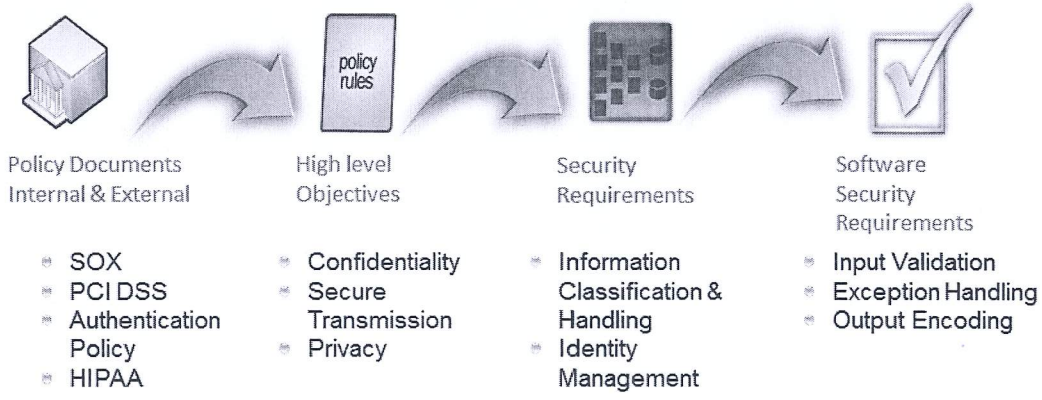

Figure 2.10 – **Policy Decomposition Process**

It starts by breaking the high level requirements in the policy documents into high level objectives which are in turn decomposed into generate security requirements, the precursors for software security requirement. As an illustration, consider the following PCI DSS requirement 6.3 example which mandates:

Develop software applications in accordance with PCI DSS and based on industry best practices, and incorporate information security throughout the software development life cycle.

This requirement is pretty high level and can be subject to various interpretations. What is the meaning of incorporating information security through the SDLC? Additionally, what may be considered as an industry best practice for someone may not even be applicable to another. This is why the high level policy document requirement must be broken down into high level objectives such as:

CFG – Configuration management

SEG – Segregated environments

SOD – Separation of duties

DAT – Data protection

PRC – Production readiness checking and

CRV – Code review

These high level objectives can be used to glean security requirements:

CFG1 – Test all security patches, and system and software configuration changes before deployment

SEG1 – Separate development/test and production environment

SOD1 – Separation of duties between development/test and production environments.

DAT1 – Production data (live sensitive cardholder data) are not used for testing or development.

PRC1 – Removal of test data and accounts before production systems become active.

PRC2 – Removal of custom application accounts, user IDs, and passwords before applications become active or are release to customers.

CRV1 - Review of custom code prior to release to production or customers in order to identify and any potential coding vulnerability.

From each security requirement, one or more software security requirements can be determined. For example the CFG1 high level objective can be broken down into several security requirements:

CFG1.1 – Validate all input on both server and client end

CFG1.2 – Handle all errors using try, catch and finally blocks

CFG1.3 – Cryptographically protect data using 128 bit encryption of SHA-256 hashing when storing it

CFG1.4 – Implement secure communications using Transport (TLS) or Network (IPSec) secure communications.

CFG1.5 – Implement proper RBAC control mechanisms.

Decomposition of policy documents is a crucial step in the process of gathering requirements and an appropriate level of attention must be given to this process.

Data Classification

Within the context of software assurance, data or information can be considered to be the most valuable asset that a company has, second only to its people. Like any asset that warrants protection, data as a digital asset needs to be protected as well.

Types of Data

Data can be primarily designated as structured data or unstructured data for the purposes of classification. When data is organized into identifiable structure, it is referred to as structured data. The best example of structured data is a database in which all of the information is stored in columns and rows. The organization of data in an identifiable structure also makes the data contents relatively more searchable by data type. Unlike structured data, unstructured data has no identifiable structure. Examples of unstructured data include images, videos, emails, documents and text. Although the examples of unstructured data may seem to have a uniform format, all data within the dataset does not necessarily contain the same structure. While some data can be stored as an image, others may be stored as a document or in an email.

Labeling

Not all data need the same level of protection as public data require minimal to no protection against disclosure. Data classification is the conscious effort to assign *labels* (a level of sensitivity) to information (data) assets, based on potential impact to confidentiality, integrity and availability (CIA), upon disclosure, alteration or destruction. This labeling can then be used for the categorization of data into appropriate buckets as depicted in *Figure 2.11*.

The Special Publication 800-18, published by NIST, provides a framework for classifying information assets based on impact to the three core security objectives, i.e., confidentiality, integrity and availability. This is highly qualitative in nature and the buckets used to classify are tied to impact as High, Medium and Low. This categorization is then used to determine security requirements and the appropriate levels of security protection by category.

The main objective of data classification is to lower the cost of data protection and maximize the return on investment when data is protected. This can be accomplished by implementing only the needed levels of security controls on data assets based on their categorization. In other words, security controls must commensurate with the classification level. For example, there is no point to encrypt data or information that is to be publicly disclosed or implementing

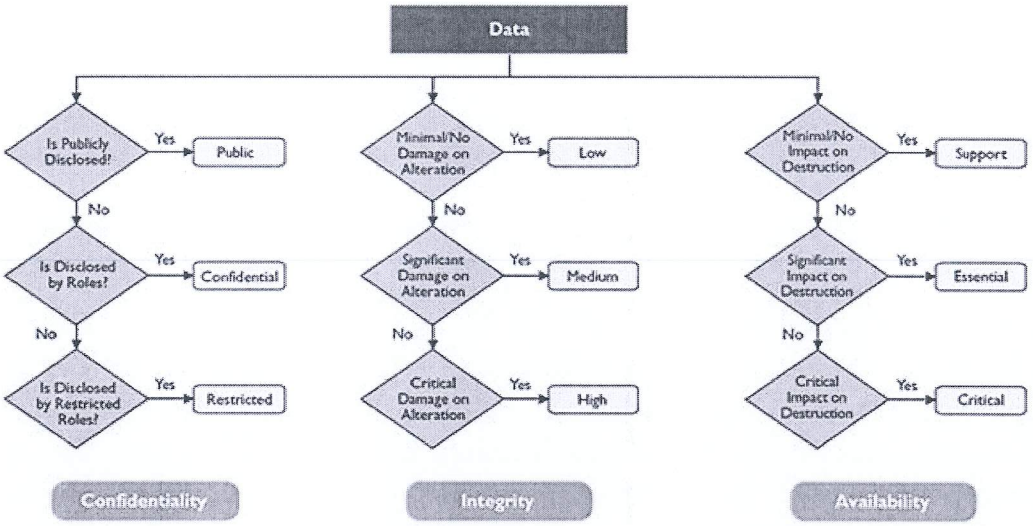

Figure 2.11 – **Data Classification Labeling**

full-fledged load balancing and redundancy control for data that has a very limited adverse effect on organizational operations, assets or individuals. In addition to lowering the cost of data protections, and maximizing ROI, data classification can also assist in increasing the quality of risk based decisions. Since the data quality and characteristics are known upon classification, decisions that are made to protect them can also be made appropriately.

Data Ownership and Roles

Decisions to classify data, who has access and what level of access, etc. are decisions that are to be made by the business owner. It is also imperative to understand that it is the business that owns the data and not the Information Technology (IT) or the information security organization. This is why the business owner is also referred to as the data owner. The business/data owner has the responsibility for the following:

- Ensure that information assets are appropriately classified.
- Validate that security controls are implemented as needed by reviewing the classification periodically.
- Define authorized list of users and access criteria based on information classification. This supports the Separation of Duties principle of secure design.
- Ensure appropriate backup and recovery mechanisms are in place
- Delegate as needed the classification responsibility, access approval authority, backup and recovery duties to a data custodian.

The data custodian is delegated by the data owner and is the individual who is responsible for the following:

- Perform the information classification exercise.
- Perform backups and recovery as specified by the data owner.
- Ensure records retention is in place according to regulatory requirements or organizational retention policy.

Data Lifecycle Management (DLM)

The term *Information Lifecycle Management (ILM)* is commonly referred to as *Data Lifecycle Management* (DLM) and the terms are used interchangeably although a distinction can be made between the two. While DLM products primarily deals with data attributes such as file types and age of files, ILM products can usually handle more complex situations, including contents within the stored data. Often when DLM is mentioned, there is a tendency to see it from purely a product perspective, it is important to recognize that DLM is not a product, but a policy based approach, involving procedures and practices, to protect data throughout the information life cycle: from the time it is created to the time it is disposed or deleted.

Data classification is usually the first and primary component of DLM. Once data is organized into appropriate categories (or tiers) appropriate controls can be applied to protect the confidentiality, integrity and availability of data.

When data is generated (i.e., created) and used (i.e., processed), transmitted, stored, and archived, appropriate protection mechanisms need to exist. Additionally, who has access to the data, the level of access (authorization rights), whether the data will be stored as structured or unstructured data and the environment (private, public, or hybrid) in which the data will be stored and used, must be determined.

Secure memory management prevents disclosure of data when data is processed. Cryptographic protection such as encryption and hashing, in conjunction with end-to-end secure communication protocols operating in the transport (e.g., SSL/TLS) or network (e.g., IPSec) layer protects data when it is transmitted. Data Leakage Prevention (DLP) technologies come in handy to protect against unauthorized disclosures when data is transmitted. Database encryption is a control that is useful to protect sensitive or private data during storage. A common type of DLM solution is Hierarchical Storage Management (HSM). The hierarchy represents different types of storage media, ranging from Redundant Array of Inexpensive Disks (RAID) systems, optical

storage, or tape, solid state drives, etc. From a future accessibility and availability of data standpoint, heuristically, more critical data that needs to be accessed more frequently for business transactions must be stored in faster media while less critical data is stored on slower media. It is also important to note that portable media are susceptible to theft and so physical security protection practices need to be in effect. Security requirements when archiving data must be considered when data is archived. The rules for data retention are determined by the corporate data retention period which must complement local legal and legislative procedures. The period of retention must be explicitly identified and enforced. However, when the data has outlived its usefulness (i.e., no longer needed for the business operations or continuity), and there is no regulatory or compliance requirement to retain it, it must be securely disposed. Secure disposal includes deletion or physically destruction of the data. Additionally, the media in which the data was stored must be sanitized

Proper implementation of data classification can be effective in determining security requirements because a one-size-fits-all security protection mechanism is not effective in today's complex heterogeneous computing ecosystems. Data classification can ensure that confidentiality, integrity, and availability security requirements are adequately identified and captured in the software specifications documentation.

Subject/Object Matrix

When there are multiple subjects (roles) that require access to functionality within the software, it is critical to understand what each subject is allowed to do. Objects (components) are those items that a subject can act upon. They are the building blocks of software. Higher level objects must be broken down into more granular objects for better accuracy of subject-object relationship representations. For example, the 'database' object can be broken down into finer objects such as 'data table', 'data view' and 'stored procedures' and each of these objects can now be mapped to subjects or roles. It is also important to capture third party components as objects in the software requirements specification documents.

A subject-object matrix is used to identify allowable actions between subjects and objects based on use cases. Once use cases are enumerated with subjects (roles) and the objects (components) are defined, a subject-object matrix can be developed. A subject-object matrix is a two-dimensional representation of roles and components. The subjects or roles are listed across the columns and the objects or components are listed down the rows. A subject-object matrix is a very effective tool to generate misuse cases. Once a subject-object matrix is generated, by inversing the allowable actions captured in the subject-object matrix, one can determine threats, which in turn can be used to determine security requirements. In a subject-object matrix, when the subjects are roles, it is referred to as a role matrix.

Use Case & Misuse Case Modeling

Like data classification, use case modeling is another mechanism by which software functional and security requirements can be determined. A use case models the intended behavior of the software or system. In other words, the use case describes behavior that the system owner intended. This behavior describes the sequence of actions and events that are to be taken to address a business need. Use case modeling and diagramming is very useful for specifying requirements. It can be effective in reducing ambiguous and incompletely articulated business requirements by explicitly specifying exactly when and under what conditions certain behavior occurs. Use case modeling is meant to model only the most significant system behavior and not all of it and so should not be considered a substitute for requirements specification documentation.

Use case modeling includes identifying actors, intended system behavior (use cases), and sequences and relationships between the actors and the use cases.

Actors may be an individual, a role or non-human in nature. As an example, the individual John, an administrator or a backend batch process can all be actors in a use case. Actors are represented by stick people and use case scenarios by ellipses when the use case is diagrammatically represented. Arrows that represent the interactions or relationships connect the use cases and the actors. These relationships may be an "includes" or "extends" type of relationship. Figure 2.13 depicts a use case and misuse case of an online ecommerce store. The customer must first create an account and sign in before placing an order. The customer need not be authenticated for searching the catalog of products. This sequence of actions is not represented within the use case itself but this is where a sequence diagram comes handy. Sequence diagrams usually go hand in hand with use case diagrams. Preconditions such as a user must be authenticated before placing an order and that they should be required to sign in again before performing authenticated user actions, can be used to clarify the scope of the use case and document any assumptions the use case author has made about the system.

From use cases, misuse cases can be developed. Misuse cases, also known as abuse cases help identify security requirements by modeling negative scenarios. A negative scenario is an unintended behavior of the system, one that the system owner does not want to occur within the context of the use case. Misuse cases provide insight into the threats that can occur against the system or software. It provides the hostile users point of view and is an inverse of the use case. Misuse case modeling is similar to the use case modeling, except that in misuse case modeling, mis-actors and unintended scenarios or behavior are modeled. Misuse cases may be intentional or accidental. One of the most distinctive traits of misuse cases is that they can be used to elicit security requirements unlike other requirements determination methods that focus on end-user functional requirements.

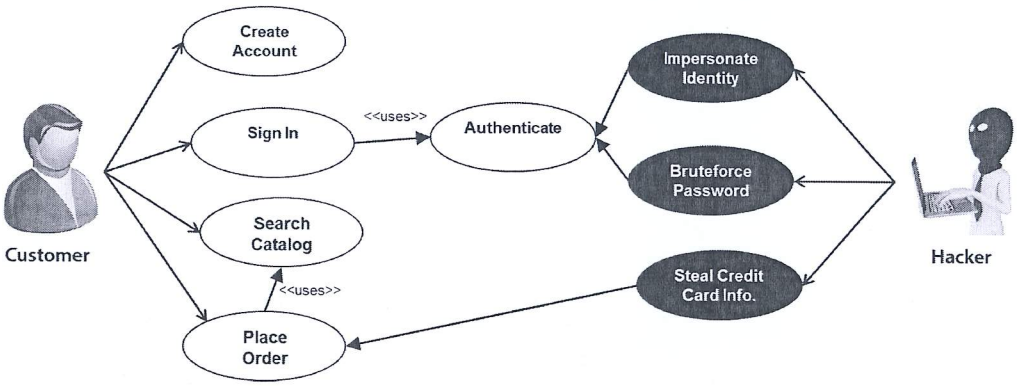

Figure 2.13 – **Example of an Online eCommerce Store Use case and Misuse case**

Misuse cases can be created through brainstorming negative scenarios like an attacker. A misuse case can also be generated by thwarting the sequence of actions that is part of the use case scenario. In our online ecommerce store example, a hacker can impersonate the identity of a legitimate customer by stealing his user name and/or bruteforce the password. A hacker can also steal credit card information of the customer and place an order, using the stolen information. In all of these scenarios, a misuse of intended behavior is what is observed. Misuse cases must not only take into account adversaries that are external to the company, but also the insider. A database administrator who has direct access to unprotected sensitive data in the databases is a potential insider mis-actor and a misuse case to represent this scenario must be specified. Auditing can assist in determining insider threats and this must be a security control that is taken into account when generating misuse cases for mis-actors that are internal to the company.

Some of the common templates that can be used for use and misuse case modeling are templates by Kulak and Guiney and by Cockburn. The Secure Quality Requirements Engineering (SQuaRE) methodology consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. The SQuaRE process model tool has been developed by the United States Computer Emergency Readiness Team (US-CERT).