**CPSC525 Assignment4**

Jiashan Li 10171607

Q1 Word count: 450

Individual tracker considered the following problem for counting queries which are answerable only for query set sizes in the range [k, n - k], where 1 < k <= n/2. The questioner knows from external sources that a given individual I, whose record is in the database, is uniquely characterized by the formula C. The questioner seeks to learn whether or not I also have characteristic a. Since COUNT (C- a) <= COUNT(C) = 1< k, if the questioner can divide C in two parts, he may be able to calculate COUNT (C. a) from two answerable queries involving the parts. This result can be extended to work for any statistical query q(C). The individual tracker is based on the concept of using categories known to describe a certain individual to determine other information about that individual. A new individual tracker must be found for each person. The general tracker removes this restriction. It employs a single formula that works for the entire database. No prior knowledge about anyone in the database is required. A general trucker is any characteristic formula T whose query set size is in the restricted sub range [2k, n - 2k]. That is, 2k <= COUNT(T) <= n-2k. Notice that q(T) is always answerable since its query set size is well within the range [k, n - k]. Obviously k must not exceed n/4 if a general tracker is to exist at a& in the worst case, k = n/4, T is a tracker if and only if COUNT(T) = n/2. By symmetry, T is a tracker if and only if p is a tracker. We might hope to secure the database by restricting the range of allowable query sets even further. The general tracker is not guaranteed to work when k > n/4, that is, when more than half the range of query set sizes is disallowed. But this does not imply that the database is secure because, corresponding to a given C, there may exist a formula T for which eqs. (There may also exist a decomposition of C for which the individual tracker works.) Even if the database could be proved secure from the general tracker when k > n/4, it may be susceptible to compromise by the method of the double tracker. A double tracker is a pair of characteristic formulas (T,U) for which k<= COUNT(T) <= n-2k, 2k<= COUNT(U)<= n-k.

Q2 Word count: 514

There are several ways for a Java Virtual Machine to protect memory. The restricted expressiveness is one of the protection features. Since JVM byte code is a strongly typed intermediate representation, unsafe instruction can not be expressed. User also is unable to forge illegal pointer and encapsulation is real. This keeps the memory safe from the attackers. Dynamic checking is also a protection feature. Untrusted code interacts with the CPU through the mediation of the JVM, and screen out or detect dangerous actions, such as null pointer dereferencing, out-of-bound array access, and illegal type-cast. In terms of access control, a Java Virtual Machine has three security components: byte code verifier, class loader and security manager. And there are typical security policies for applets such as applets do not get access to the users’ file system, applets cannot obtain information about the user’s name, email address and machine configuration, applets may make outward connections only back to the server they came from, applets can only pop up windows that are marked ‘untrusted’, applets cannot reconfigure the JVM, e.g. by substituting their own classes for system classes. The byte code verifier provides basic well-formedness checks and static type checking. And because JVM byte code is an unstructured language, type checking involves a non-trivial data flow analysis. This makes sure the class file is in the proper format, the stacks will not. overflow, there is no illegal data conversion between types, all references to other classes are legal and so on. The class loading is delayed as much as possible to reduce memory usage and improve system response time. When a class loader is being instantiated it reads in the byte code, defines the class, and assigns it a protection domain. Link-time checks are performed by the JVM to maintain type safety. Every JVM has a class file verifier which makes sure that ‘untrusted’ classes are safe. The security manager is invoked at runtime to check whether a process requesting access to a protected resource has the required permissions. The access controller implements a uniform access decision algorithm for all permissions. To compute the granted permissions, the access controller examines the given execution context, i.e. the execution stack, and performs a stack walk. Each method on the stack has a class and each class belongs to a protection domain. The protection domain defines the permissions granted to this method. The basic access control algorithm computes the intersection of granted permissions for all methods in the execution context. It grants access if all methods on the stack have been granted the required permission. The JVM in the web browser enforces security in a layer above the operating system. Once a user has access to the layer below the security mechanisms, e.g. by running application other than the browser, all bets on the integrity of the protection system are off. Furthermore, the Java platform is not secure simply because Java is designed to be fixed. Not all software security issues are addressed by the Java type system. For example, it is up to developers to deal with race conditions.

Q3 Word count: 434

Let O represent objects, let S represent subject, let R represent right. The state of a system is the collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system. The subset of this collection that deals with protection is the protection state of the system. An access control matrix is one tool that can describe the current protection state. Consider the set of possible protection states P. Some subset Q of P consists of exactly those states in which the system is authorized to reside. So, whenever the system state is in Q, the system is secure. When the current sate is in P-Q, the system is not secure. Our interest in representing the state is to characterize those states in Q, and our interest in enforcing security is to ensure that the system state is always an element system from entering a state in P-Q is the function of a security mechanism. The access control matrix model is the most precise model used to describe a protection state. Each entry M[s,o] belongs to R is the set of rights that subject s is authorized to exercise on object o. Every column corresponds to an object. Every row corresponds to a subject. If we want to delegate a permission by imposing a delegation depth, we can let M1[S1,O] = M2[S2,O] = … = Mn+1[Sn+1,O], n = delegation depth, this means if the first person has a right on an object, and based on the delegation depth rule, he can pass the right to n people, so n+1 people have the right in total. As the system changes, the protection state changes. When a command changes the state of the system, a state transition occurs. Very often, constrains on the set of allowed states use these transition occurs. Very often, constrains on the set of allowed states use these transition occurs. Very often, constraints on the set of allowed states use these transitions inductively; a set of authorized states is defines, and then a set of operations is allowed on the elements of that set. The result of transforming an authorized state with an operation allowed in that state is an authorized state. By induction, the system will always be in an authorized state. Hence, both states and state transitions are often constrained. We use γ to represent original state and γ’ to represent new state. If somebody has the right r, he can transfer the right to n people if the delegation depth is n. So for all that n subjects have Mγ’[s,o]<-Mγ[s,o]U{r}.

Q5 Word count: 267

In the Ring Policy of Biba model, no write up will happen. S can write to O if i(o)<=i(s), where the higher i(o) is the more accurate and/or reliable o is, the higher i(s) is the more confidence one has that s will process data correctly. Secure invocation could also happen and s1 can execute s2 if i(s2)<=i(s1). In the Strict Integrity Policy, no read down will happen, that means, s can read o if i(s) <i(o). Also no write up, secure invocation will take place. No write up and Secure invocation will also happen in the Low-Water-Mark Policy as same as the first two policies. However, there is a new policy called Read downgrade. If s reads o, then i’(s)=min(i(s),i(o)), where i’(s) is the subject’s integrity level after the read. The integrity level of a subject will become lower and lower over time: the low water mark of all the object it has read. Consequently, a subject can access fewer and fewer objects over time. In terms of flexibility. The Low-Water-Mark Policy is based on the premise that the integrity level of the subject is dynamic and will change based on its previous behavior. The integrity level of the subject will be determined by the integrity level of the most recently accessed object. The integrity level of the objects in the system will not change. The Ring Policy increases the flexibility of the system by allowing observation of objects at any level. The Strict Integrity Policy does not change the integrity level of a subject. Access requests to levels which exceed the subject’s level are denied.

Q7 Word count: 235

Least Privilege is supported because RBAC can be configured so only those permissions required for tasks conducted by members of the role are assigned to role. The principle of least privilege requires that a user be given no more privilege than necessary to perform a job. Ensuring least privilege requires identifying what the user's job is, determining the minimum set of privileges required to perform that job, and restricting the user to a domain with those privileges and nothing more. By denying to subjects transactions that are not necessary for the performance of their duties, those denied privileges cannot be used to circumvent the organizational security policy. Through the use of RBAC, enforced minimum privileges for general system users can be easily achieved. Separation of duties is achieved by ensuring that mutually exclusive roles must be invoked to complete a sensitive task. RBAC mechanisms can be used by a system administrator in enforcing a policy of separation of duties. Separation of duties is considered valuable in deterring fraud since fraud can occur if an opportunity exists for collaboration between various job related capabilities. Separation of duty requires that for particular sets of transactions, no single individual be allowed to execute all transactions within the set. Data abstraction is supported by means of abstract permissions such as credit and debit for an account, rather than the read, write, execute permissions typically provided by the operating system.