Insider Abuse Comprehension through Capability Acquisition Graphs

Sunu Mathew, Shambhu Upadhyaya, Duc Ha, Hung Q. Ngo
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY, U.S.A.
Email: {smathew2, shambhu, ducha, hungngo}@cse.buffalo.edu

***Abstract*— Insider attacks constitute one of the most potent, yet difficult to detect threats to information security in the cyber-domain. Malicious actions perpetrated by privileged insiders usually circumvent intrusion detection systems (IDS) and other mechanisms designed to detect and prevent unauthorized activity. In this paper, we present an architectural framework and technique to aid in situation awareness of insider threats in a networked computing environment such as a corporate network. Individual actions by users are analyzed using a theoretical model called a Capability Acquisition Graph (CAG) to evaluate their cumulative effect and detect possible violations. Our approach is based on periodic evaluation of the privileges that users accumulate with respect to critical information assets during their work-flow. A static analysis tool called ICMAP is used to periodically construct CAGs which are then analyzed to uncover possible attacks. The process is demonstrated by considering an information process cycle from the real-world.**

Keywords: Capability Acquisition Graph, insider threat, situation awareness.

1. Introduction

Insider attacks in the cyber domain [1] constitute serious threats to information assurance especially in critical corporate and intelligence community (IC) environments. Security mechanisms such as IDSs, firewalls and access control mechanisms are designed to prevent *unauthorized* accesses, and hence are unable to detect *privilege abuse* by authorized entities. Insider attacks may result from a combination of actions among several users (*insider collusion*) and information entities, and as such, may require macro-level analysis of these interactions. Modeling the security of information assets requires consideration of factors such as network topology, location of information assets, vulnerability information, and connectivity and reachability among users/data. Another non-trivial issue is the granularity of data that is to be used in modeling for security evaluation – system calls, user commands or network data. Security audit and network hardening against insider threat is further complicated by the fact that the threat arises from privileged users that already have access to protected information. Hence an insider threat situation awareness and mitigation technique must fuse information about data location, access and user actions, and also incorporate metrics that can evaluate the relative vulnerability of information assets to insider attack.

In this paper, we present the use of Capability Acquisition Graphs (CAG) [2] and the associated tool ICMAP [3] for the perception and comprehension of attacks that arise from privilege abuse on the part of insiders in typical networks such as corporate cyber-networks. We describe how the approach can alert against successive accumulation of privileges by users in the process of mounting an attack. The approach is analyzed in the context of realistic insider attack scenarios that can arise in a typical work-flow involving different users. The rest of this paper is organized as follows: Section 2 presents a technical overview of insider threat analysis using CAG and ICMAP, and Section 3 presents an analysis of the process cycle in an intelligence process with a view to understanding the insider threat. Our approach to insider threat situation awareness is described in Section 4, followed by comparison with related work in Section 5. Section 6 concludes the paper by identifying some directions for future research.

2. Modeling Insider Threat

In this section, we define the concepts of insiders and insider threats, and describe how this can be modeled using Capability Acquisition Graphs (CAG). ICMAP, an implementation of the CAG concept, is also described.

2.1. Insiders and the Insider Threat

Insider threat is considered to be one of the most difficult and critical problems in computer security [4]. The insider threat is especially critical in financial and military networks because of the possible large-scale damage. A RAND workshop [1] defines an insider as "someone with access, privilege, or knowledge of information systems and services." The same report defines the insider threat problem as "malevolent (or possibly inadvertent) actions by an already trusted person with access to sensitive information and information systems."

Although insiders are capable of carrying out various attacks such as *masquerading* and *sabotage*, *privilege abuse* is considered to be the most threatening. This is the most difficult category to characterize and detect, since the insider *already has the privileges* needed, yet uses them in such a way that it constitutes *abuse*. Several variations of insider privilege abuse are possible:

- Information Leak: An insider with access to privileged information can take actions so that such information becomes accessible to users without necessary privileges.

- Information Gathering or Snooping: An insider with some level of privilege can try to gather information that is not relevant to his/her job function, for espionage or other purposes.
- Data Correlation and Information Aggregation: An insider can seek to refine his/her information or awareness about some entity by selectively targeting data that is relevant to it (e.g., querying multiple databases trying to obtain personal information about an individual).

Insider threat research currently suffers from a lack of proper problem formalization. We aim to define and develop solutions for a subset of insider threat, viz. privilege abuse attacks. The detection of insiders with respect to a set of policies is counter-intuitive since an insider is always defined *relative* to a set of policies. In other words, given a set of policies, there will *always be a possibility of insider attack*. This notion is encapsulated in the definition of insider provided in [5] – "An insider with respect to rules R is a user who may take action that would violate some set R of rules in the security policy were the user not trusted. The insider is trusted to take the action only when appropriate, as determined by the insider discretion."

2.2. Capability Acquisition Graphs (CAG)

An information-centric approach to modeling the insider threat in a typical network was presented in [2] and [3]. The *Capability Acquisition Graph (CAG)* model and the associated tool *ICMAP* allow analysts to intuitively model information about the location and reachability of information assets on a network in a manner reflecting the physical layout of the network (unlike other models such as attack graphs [6], [7]). We describe the model briefly:

Definition 1: A capability acquisition graph is a tuple represented by:

$$CAG = (V, E, K, V_0, V_S, \pi, \delta) \quad (1)$$

V is a set of nodes representing physical entities such as hosts, firewalls, user accounts. E is a set of edges; an edge connects two nodes if one node can be reached from the other. K is a set of tokens representing system information or individual information such as a password. V_0 is the set of start nodes from where an attack can be launched; the set V_0 can be adjusted to model the skill-set of an attacker. V_S is the set of target nodes in the logical graph that an attacker intends to compromise. The function $\pi : V \rightarrow K$ assigns tokens to nodes, e.g., a database node may have records as tokens. The function $\delta : E \rightarrow K \times N \times N$ represents the edge attributes, consisting of token challenges (costs of traversing the edge).

A CAG can be viewed as an abstract representation of a user's walk in a network. A user starts from a particular node in the graph with certain tokens (knowledge). From the starting node, the user chooses an edge, $e(u, v) = (token, min, max)$, to move to an adjacent node. If the $token$ is already present in his set of knowledge, he incurs a cost of min otherwise he incurs a cost of max. If V' is the current set of visited vertices, then the cost of visiting a new vertex $v \notin V'$ is the minimum cost edge (u, v) for all $u \in V'$. The cost of an attack sequence or attack trail $(v_1, v_2, \ldots, v_n)$ is the sum of the costs of visiting a new vertex from the set of already-visited vertices.

A reasonable assumption regarding attacker behavior is that he aims to minimize his cost of reaching targets by choosing edges with simple token challenges. Security analysts aim to harden the network by assigning edge token challenges so that critical information is protected. By enumerating the least-cost paths, likely attack paths can be identified and the network can be secured by eliminating these paths or by placing sensors such as IDSs along them.

Specification of the CAG model begins by identifying the scope of the threat, which may range from a small portion to the entire organization under consideration. The physical layout of the entities and critical information assets are identified – the size of the resulting model is a polynomial function of the input information size. However, the problem of determining the cost of least resistance in a CAG is NP-Hard [8]. In fact, the problem is not even approximable to within $2^{(\log n)^{1-\delta}}$ where $\delta = 1 - \frac{1}{\log \log^c n}$ for any $c < 1/2$. Therefore, finding a least cost attack in an efficient manner is not possible unless P = NP.

A greedy heuristic approach involving a one-step lookahead may be used to identify an optimal walk [2], [8]. Sometimes, even if a shorter path to a goal exists, an attacker might avoid it believing that sensors might be placed along the path. Therefore, the greedy heuristic approach has to be run multiple times to identify the k-best paths instead of one optimal path. CAGs can also represent social engineering channels (e.g., telephone lines when identifying insider abuse paths).

2.3. Information-Centric Modeler and Auditor Program (ICMAP)

The architecture of ICMAP, an information-centric modeling and analysis tool based on CAG, is depicted in Figure 1. It takes the physical network topology and information about vulnerabilities in network services as external inputs, and combines them with network translation rules and cost rules to obtain the CAG. A 'physical graph' representing a network consists of hosts, routers, firewalls, network services such as *ssh*, *ftp*, *http*, *nfs* and databases. A host contains the host id, user accounts, network services, vulnerabilities and critical files (called "jewels"). In order to build the CAG, for each host, ICMAP draws the user account nodes, the service nodes, the vulnerability nodes, and the jewel nodes. A user (or a malicious insider) either connects to a service remotely or logs in from the console. Once the user gains access to a host he uses the network resource and connects to another host, uses the file system resource and edits files, exploits vulnerabilities to escalate his privileges, or uses the cpu resource on the host to execute programs, check email, browses and so on. To represent the above activities, edges (with their token challenges) are drawn entering the user accounts. The token challenges are marked on the edges, if the token is known, then traversing the edge incurs a cost of LOW, otherwise a cost of $HIGH$ is incurred. Edges marked

"0" do not have a token challenge, so they always incur a cost of LOW. From the user accounts there exists a zero-cost transition to the host service, and from the host there exist transitions to other accounts in the network. We also add zero-cost transitions from the root account to other accounts in the same host to express the fact that the root can become any user. Once a user gets to the host, vulnerabilities in the services can be exploited; thus edges are drawn from the services to their vulnerabilities. The tokens in the vulnerability node can be used to escalate privileges (e.g., become root). Finally, edges exist from the user accounts and network services (e.g., `ssh` and `ftp`) to the file system (e.g., `nfs`) of the host and from the file system to the jewels.

It is important to mention that the automatic graph conversion (from physical to logical) is intended to reduce the work of an analyst, not to limit it. After the conversion, an analyst can still perform various adjustments to the logical graph (e.g., add/remove relationships, tokens and change the costs). Adjustments to the physical graph at this step are also automatically updated to the CAG. Further details on ICMAP and CAG construction are available in [3]. The logical graph (CAG) for a subnet consisting of an *ssh* server, *ftp* server and a firewall is depicted in Figure 2.

Once the CAG is constructed, various heuristics, e.g., 1-step, k-step (constant k) and n-step lookahead techniques, can be used to find an optimal path from a source to a destination without having to enumerate all possible paths. Also, using combinations of source and destination pairs, it is possible to identify the best locations to position network sensors.

Two separate analyses can be performed on a CAG to refine the threat assessment. The first is sensitivity analysis where different cost assignments are used to identify the optimal cost assignment that results in attack paths that are similar to known attacks. The second technique is to perform a defense-centric analysis where sensors are placed along the paths of least resistance to help prevent network assets from being compromised. The cost assignment is refined based on these two analyses.

3. Insider Threat in the Information Process Cycle

In this section, we take a closer look at the activities within a specific information process in order to identify generic observations that may be useful for mitigating insider threat using formal models such as CAG. As a case study, we consider an Intelligence Production process (described in [1]) that may come under various attacks. The process life cycle consists of the following stages:

- **Requirement:** Statement of need by a consumer in the form of a formalized request. There is a high potential for insider threat, and vulnerabilities include activities such as *requirements modification*.
- **Collection:** This process involves the collection of raw data. The insider threat potential here is less likely and involves activities such as degrading data and denial-of-service (DoS).
- **Processing and Exploitation:** Involves selecting/filtering the data into usable form.
- **Analysis and Production:** Analysis is the process of transforming information to knowledge while Production formalizes the knowledge in the form of a document or product. This phase involves tools such as document management systems and has considerable potential for malicious insider activity.
- **Dissemination:** Dissemination is concerned with distributing information to authorized consumers (users). This may include both electronic means such as *email* and non-electronic such as hard-copy documents. The potential for insider attack and collusion is perceived to be high.
- **Consumption:** This involves use of the produced information by users. Vulnerabilities here include exfiltration, leak and misuse. The potential for the insider threat is seen to be high.

A systematic approach to mitigating the insider threat within this process cycle begins by identifying classes of activity or 'meta-attacks' that may indicate malicious activity. For example, the following 'meta-attacks' are identified within cycle stages which could indicate possible insider attempts at information leak [1]:

- **Access:** Authorized accounts, Orphan account, Unlocked, Unattended terminals, Document control, File permissions, Need-to-know violations, Password guessing, Privilege escalation, Accidental/incidental access etc. (Non cyber methods may be social engineering, shoulder surfing
- **Reconnaissance:** Web/file browsing, database searches, Unusual searching, stealthy scanning
- **Entrenchment and Exploitation:** Download, Media import, email virus/trojan, keystroke logger, Import published attack, install unauthorized software, install sensor/bot, sabotage patch system, replace device drivers/analysis tools
- **Extraction and Exfiltration:** Printing/copying, manual classification downgrade, removable media, masqueraded media, wireless usage, steganography, duplicate database, Log file/backup
- **Communication:** Standard encrypted email, simple coded messages, wireless usage, custom encrypted email, steganography, covert channels
- **Manipulation:** Altering authorized information, upgrading classification, database modification, corrupt protections – virus, corrupt infrastructure
- **Counter intelligence:** Unusual file deletion, Blocking admin access, search case files, disk erase, modify case files, modify audit logs, normal drift, replace device drivers, analysis tools
- **Other activities:** Pornography, gambling

Some of the cyber-activities identified above explicitly indicate malicious activity while others may be more subtle raising the possibility of covert insider activity. The effects

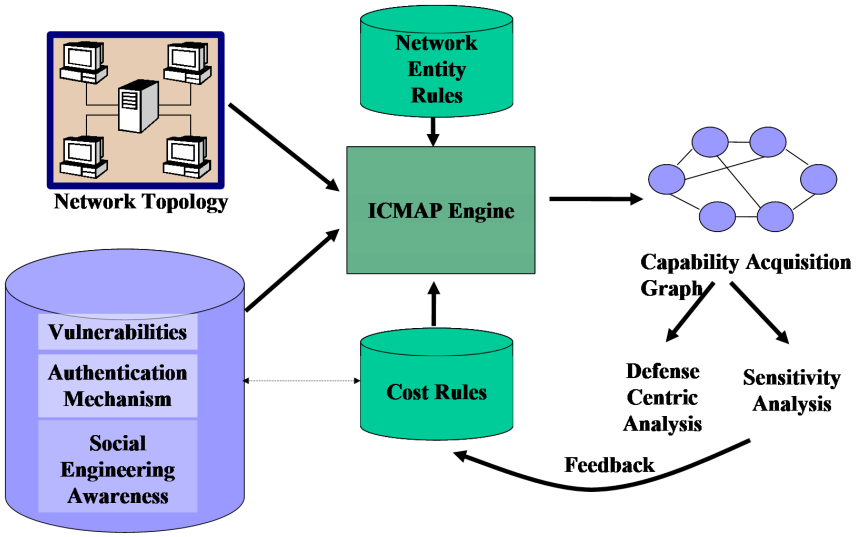

Fig. 1. ICMAP framework.

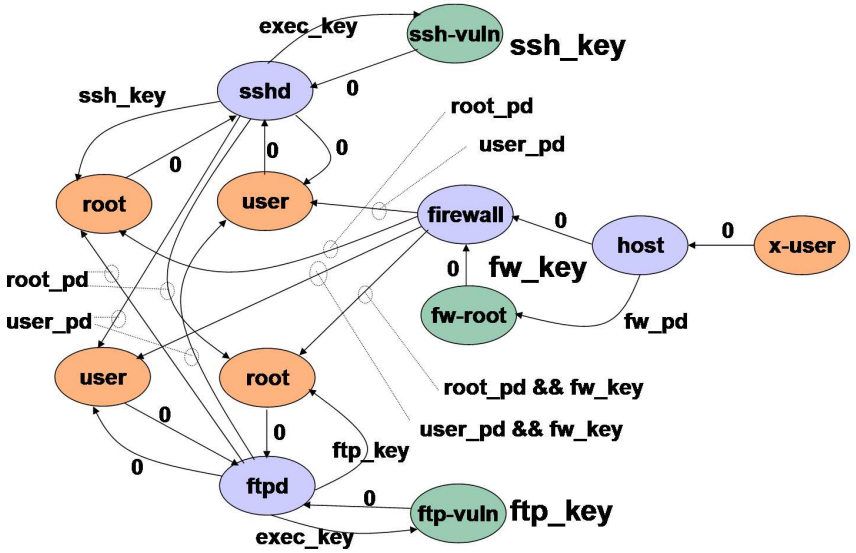

Fig. 2. Logical graph of a subnet.

of these meta-attacks may be broadly classified as affecting **Confidentiality**, **Integrity** and **Availability/Denial of Service** or constituting an **Enabling Action for another attack**.

3.1. Understanding Insider Attacks: An Example

We study the mitigation of insider attack scenarios by presenting and analyzing a multistage collusion attack involving interactions between an *Admin* with *Top-Secret* access privileges, a *Programmer* with *Classified* privileges and a *Secretary* with privileges to access only *Public* documents in a corporate network. We try to interpret the attack stages in the light of the process life cycle, 'meta-attacks' and effects presented above.

- **Stage 1:** Life Cycle Stage: *Consumption*
  Meta-Attacks: Admin – *Exfiltration*, Programmer – *Access*
  Effects: *Enabling Action*, *Confidentiality*
  Description: The *Admin* initiates the attack by checking out a *Top Secret* document and transferring information to a classified document (lower classification). This transfer can be very minimal (like the summary of the financial outcome of a transaction, or some crucial part of a product blueprint).

- **Stage 2:** Life Cycle Stage: *Requirement*, *Consumption*
  Meta-Attacks: Programmer – *Exfiltration*, *Manipulation*, Secretary – *Access*

Effects: *Enabling Action*, *Confidentiality*
Description: The *classified* document is rarely required by the programmer. After a few weeks/months (and even years in a military context, a project requirement necessitates that the programmer access the *classified* document. After performing the required work function, the programmer transfers the information to a *public* document (or at least some document which the secretary has access to, say, his own payroll information). This stage illustrates two major points:

- The programmer does not access/check-out the classified document only for the purpose of leaking information; in fact, given the context, he had to access the document for the project.
- All information transfers would be seen as legitimate and required for the job function.

This is an example of illegal information transfer masked by legitimate contextual requirements.

- **Stage 3:** Life Cycle Stage: *Consumption*
Meta-Attacks: Secretary – *Exfiltration*, *Communication*
Effects: *Confidentiality*
Description: The secretary then accesses the *public* document (again, after a considerable period of time and under a suitable context, say to process the programmer's payroll information for updating records) and publishes it or transfers it to some outside collaborator (under the guise of sending, say, a press release which is again *public* information.

Analysis of the attack stages indicates that as a result of actions taken by the *Admin* in Stage 1, the contents of the *Top-Secret* document become available to users (i.e., the *Programmer*) with lower classification or privilege levels. The activity of the *Admin* leading to alteration of the privilege set of the *Programmer*, and the increased privilege accumulation on the part of the *Programmer*, if detected and analyzed by security analysts, can indicate *insider collusion* between the two users leading to possible information leakage. A similar set of interactions occurs in Stage 2 between the *Programmer* and the *Secretary*. Stage 3, in which the original *Top-Secret* information shows up in the public domain, is when the information leak is usually detected. Monitoring user privilege accumulation with respect to sensitive information over time can enable effective forensics after an attack has been detected, and even help to mitigate attacks by raising suitable alerts when unauthorized privilege accumulation becomes apparent. This concept forms the basis of our insider abuse comprehension technique described next.

4. Detecting Insider Abuse by temporal CAG analysis

Our approach to insider threat detection is based on the idea of evaluating *user intent*. Our evaluation is based on a CAG based security analysis that tracks the cost of users' traversal to "jewels". Any work-flow activity that results in high-value assets being easily accessible to unauthorized users may be indicative of insider attack and must be analyzed by security analysts as a precaution. We elaborate this technique in this section.

ICMAP generates CAG diagrams for a network once given required input information. The CAG can be used to perform static analysis of the security state of the network – security personnel can ensure that critical "jewels" are well protected. Actions (e.g., copying information from one document to another) by users as part of normal activity can result in a gradual variation in the security state of the network – cheaper cost alternative paths to once well-protected "jewels" might emerge and the privileges associated with different users and user groups might change. Hence, in the absence of privilege monitoring, malicious insider activity may proceed undetected resulting in security breaches such as exfiltration or leakage of sensitive information.

4.1. Checkpoint Based CAG Analysis

Tracking user activity with respect to "jewels" is essential for detecting and mitigating malicious insider activity. A combination of event sensors can be used for this purpose – IDSs such as *Snort* [9], *Dragon* [10], file-integrity checkers such as *Samhain* [11] and application-specific document management systems [12] can be used to provide fine-grained indications of user activity associated with "jewels" such as sensitive documents.

Online security analysis based on event activity may, however, not be practically feasible. This is because even a small number of user actions (for example, moving or replicating one document or "jewel" from one node to another) can result in a drastic change in the logical graph or CAG. This, combined with the computationally expensive nature of CAG analysis [8], means that periodic construction and analysis of CAGs is the preferred approach. To this end, we propose the approach depicted in Figure 3, which is based on CAG *checkpoints*. Physical network configuration details are used by the ICMAP tool to generate an initial CAG. Events from sensors are logged continuously and CAG updates occur periodically at well defined checkpoints. The events logged after the last checkpoint are used to decide how to reconstruct or update the CAG. Analysis of low-cost paths to "jewels" can raise alerts if certain paths have below-threshold costs. Analysts can then study event logs for suspicious activity and even manually adjust model parameters for increased sensitivity. Periodic checkpointing of CAG states is also useful for forensic evaluation once a security breach has occurred:

- The source(s) of the breach, or 'candidate attackers' are likely to be users that had cheaper paths to the data in question, especially if such privileges were acquired recently.
- The time-frame and exact sequence of actions can be identified by focusing on events logged between relevant CAG checkpoints – for example, between two checkpoints that indicate the greatest change in privileges for 'candidate attackers.'

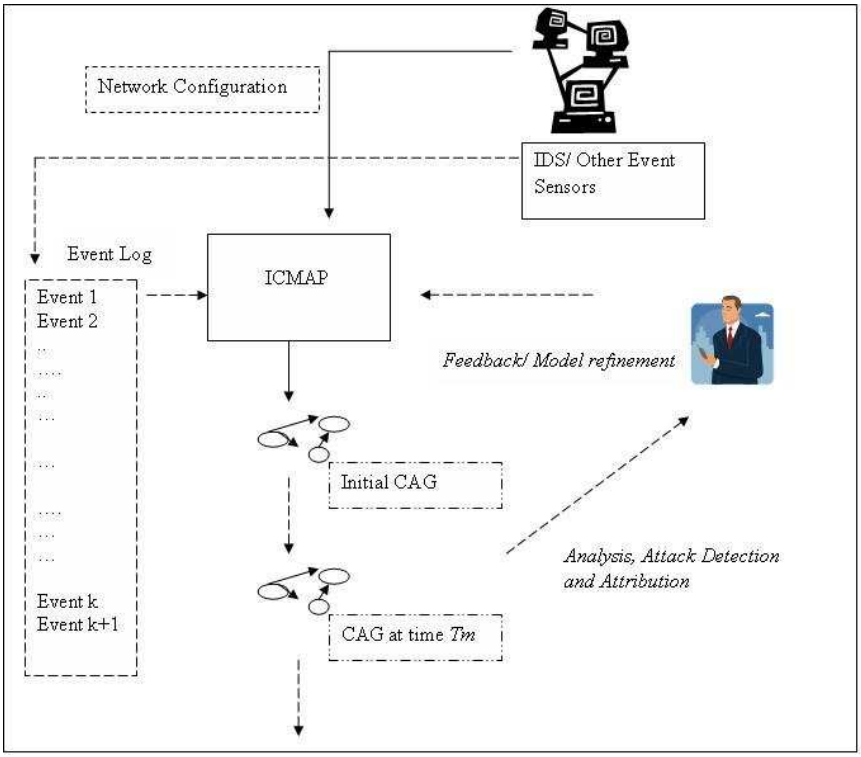

Fig. 3. ICMAP generates Capability Acquisition Graphs based on sensor events for network security analysis

- Insider collusion, or cooperative activity among multiple users with malicious intent, can also be detected by analyzing events that contributed to the security breach.

Thus, ICMAP can aid forensic efforts by providing attribution information (who is the likely insider) based on comparisons between successive CAG constructions, and also predict attacks (given the new CAG and its cost analysis, what data is vulnerable to attack by whom?). We illustrate the application of ICMAP to insider threat detection by presenting a real-world example.

4.2. CAG modeling: A real-world example

Consider a scenario in which documents of different classifications and three types of users are defined with appropriate privilege levels. The attack entails downgrading a document successively by people with the right privileges with the ultimate goal of leaking top-secret information to the public. In this scenario, each domain requires certain capabilities to access (represented as keys).

In Stage 1 (Figure 4), the top-secret file *UnderCover.doc* lies in the *Top-Secret* domain and is only accessible by the Administrator. The actual content of the file is denoted as "DB". That information is currently not accessible to the Programmer or any other user.

In Stage 2 (Figure 5), the Administrator modifies and transfers the content of *UnderCover.doc* to *Classified.doc* in the Classified domain. The *Classified.doc* file now has a new "jewel" information "DB" associated with it. The arrow from the *Top-Secret* domain in Figure 5 depicts the operation of the admin, resulting in a new state of the network. From this state, ICMAP can now determine the various low-cost access paths and sequences to the "DB" information.

In Stage 3 (Figure 6), the Programmer copies the content of *Classified.doc* to the normal file *Salary.txt* in the public domain. Consequently, the Secretary now has access to the undercover agent list. Again, when provided the action taken by the Programmer (arrow in Figure 6), ICMAP can construct the new network state and determine how the jewel "DB" is accessible to various attackers including the Secretary.

Since optimal analysis of the CAG takes exponential running time, it becomes necessary for ICMAP to depend on good heuristics [2], [8] in the practical case. ICMAP can be configured to return the top-k (k is a configurable parameter) attack trails or scenarios for security analysis. We see that even for relatively large CAGs (200-300 nodes), the running time of the heuristics is a few minutes [2]. Hence it is realistic for ICMAP to reconstruct and analyze the CAG of the network at periodic intervals (e.g., a few times a day). Further details on ICMAP's CAG analysis are available in [8] and [3]. Additional empirical tests with realistic networks is necessary to obtain further insight into the actual running time and performance of the heuristics.

5. Comparison with Related Work

Network modeling to detect attacks has been well studied in the form of *attack graphs* [7], [6]. Attack graph construction

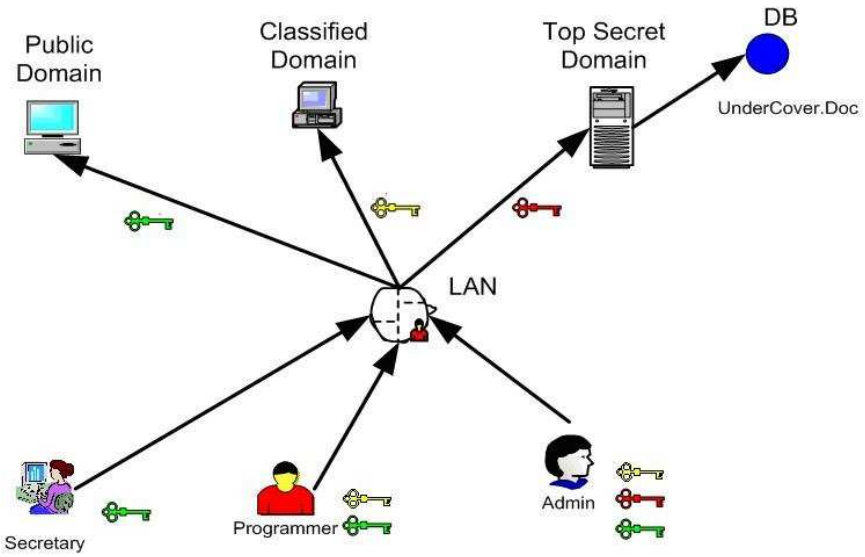

Fig. 4. Document compromise – Stage 1

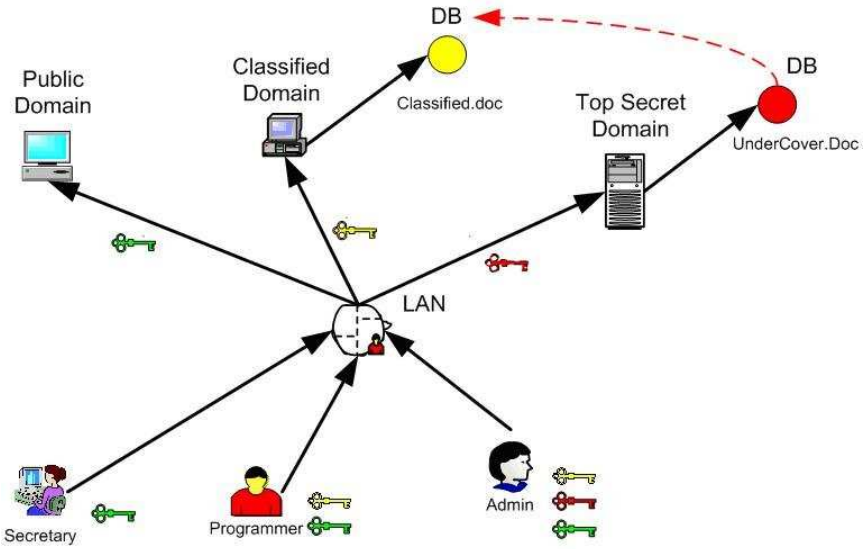

Fig. 5. Document compromise – Stage 2

from real network configurations is labor intensive and error-prone; it is also not clear if system variables can be modeled systematically for realistic-sized networks. Some attack-graph approaches used model-checking [6], which in many cases leads to the combinatorial explosion problem [3]. Our approach, in comparison, is intuitive since it reflects the physical layout of the network and "jewels" are easily identified by network analysts. The size of our model is a polynomial function of input size [2], and thus is practical as a visual aid for security analysis in realistic networks.

Addressing the insider threat in cyber-environments has also been the subject of much research. A system dynamics based approach to modeling the insider threat was presented in [13]. Using Hidden Markov Models (HMM) for insider threat detection in environments where the work-cycle consists of well-defined stages is proposed in [14]. Specific aspects of the problem such as maintaining integrity of files from insider attack is the focus of efforts such as [15]. A Bayesian-network model for insider threat detection by analyzing user behavior is presented in [16] and a semantics-based approach to protecting documents in an intelligence environment is presented in [17]. Use of attack decomposition trees and *attack vs. defense* matrices for insider threat defense is advocated in [18]. Our approach is complementary to these efforts, and in many cases is more general, since varied resources (e.g., documents, keys) can be modeled. Also, our approach can incorporate non-cyber aspects such as social-engineering interactions.

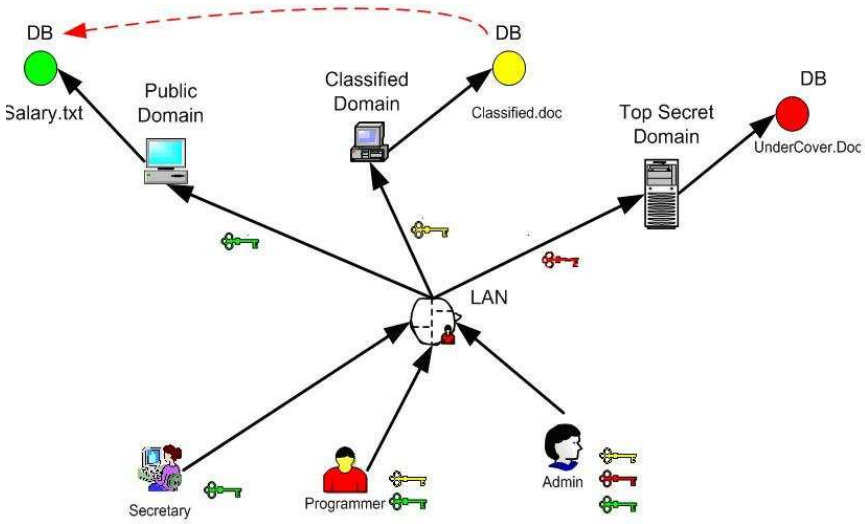

Fig. 6. Document compromise – Stage 3

6. Conclusion

In this paper, we have presented an approach to insider threat detection and mitigation that is based on monitoring privilege accumulations by users on a network. Limiting the accessibility of users to critical assets is important, but complex interactions that occur as a part of normal work-flow can provide a cover for hard-to-detect attacks such as *insider collusion*. We use a formal model, the CAG, as a means for security evaluation, and perform such evaluation at periodic checkpoints. Such an approach can aid in effective forensics as well as in attack mitigation by indicating both vulnerable data and potential attackers. The quality of the attack scenarios generated by the ICMAP heuristics, the amount of relaxation that can be safely tolerated with respect to user privilege accumulation, the frequency of CAG checkpoints, and the selection of optimal CAG parameters to protect critical assets are some open research questions concerning the approach. Further work is needed to implement the proposed solution as an integrated system supported by a comprehensive set of event sensors in order to answer these practical questions.

References

[1] R. H. Anderson and R. Brackney, "Understanding the Insider Threat: Proceedings of a March 2004 Workshop," *RAND National Defense Research Institute*, 2004.

[2] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya, "Towards a theory of insider threat assessment," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005, Yokohama, Japan)*. IEEE, 2005.

[3] Duc Ha, Shambhu Upadhyaya, Hung Ngo, Suranjan Pramanik, Ramkumar Chinchani and Sunu Mathew, *Insider Threat Analysis Using Information Centric Modeling*. P. Craiger and S. Shenoi (Eds.), Advances in Digital Forensics III, Springer, Boston, 2007.

[4] M. Bishop, "The insider problem revisited," in *NSPW '05: Proceedings of the 2005 workshop on New security paradigms*. New York, NY, USA: ACM Press, 2005, pp. 75–76.

[5] ——, "Insider is relative," in *NSPW '05: Proceedings of the 2005 workshop on New security paradigms*. New York, NY, USA: ACM Press, 2005, pp. 77–78.

[6] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated Generation and Analysis of Attack Graphs," in *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA., May 2002. [Online]. Available: http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/calder/www/sp02.html

[7] S. Jha, O. Sheyner, and J. Wing, "Two Formal Analyses of Attack Graphs," in *15th IEEE Computer Security Foundations Workshop (CSFW'02)*, Cape Breton, Nova Scotia, Canada, 2002, pp. 49–63.

[8] R. Chinchani, D. Ha, A. Iyer, H. Ngo, and S. Upadhyaya, "On the hardness of approximating the min-hack problem," *Journal of Combinatorial Optimization*, vol. 9, no. 3, pp. 295–311, 2005.

[9] "Snort intrusion detection sensor - http://www.snort.org." [Online]. Available: http://www.snort.org

[10] P. Mueller and G. Shipley, "Dragon Claws its Way to the Top, Network Computing," 2001.

[11] "Samhain file-integrity checker - http://www.la-samhna.de/samhain." [Online]. Available: http://www.la-samhna.de/samhain/

[12] S. Pramanik, V. Sankaranarayanan, and S. J. Upadhyaya, "Security policies to mitigate insider threat in the document control domain," in *ACSAC*, 2004, pp. 304–313.

[13] I. J. Martinez-Moyano, E. H. Rich, S. H. Conrad, and D. F. Andersen, "Modeling the emergence of insider threat vulnerabilities," in *WSC '06: Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 562–568.

[14] P. Thompson, "Weak models for insider threat detection," in *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III. Edited by Carapezza, Edward M. Proceedings of the SPIE, Volume 5403, pp. 40-48 (2004).*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, E. M. Carapezza, Ed., vol. 5403, Sept. 2004, pp. 40–48.

[15] A. Rocke and R. DeMara, "Mitigation of Insider Risks using Distributed Agent Detection, Filtering and Signaling," *International journal of Network Security*, vol. 2, no. 2, pp. 141–149, 2006.

[16] Costa, Paulo C. G.; Laskey, Kathryn B.; Alghamdi, G.; BarbarÃ¡, Daniel; Shackelford, Thomas; Mirza, Sepideh; and Revankar, Mehul, "Dtb project: A behavioral model for detecting insider threats," in *2005 International Conference on Intelligence Analysis*, 2005.

[17] B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, and A. P. Sheth, "An ontological approach to the document access problem of insider threat." in *ISI*, ser. Lecture Notes in Computer Science, P. B. Kantor, G. Muresan, F. Roberts, D. D. Zeng, F.-Y. Wang, H. Chen, and R. C. Merkle, Eds., vol. 3495. Springer, 2005, pp. 486–491.

[18] V. N. L. Franqueira and P. A. T. van Eck, "Defense against insider threat: a framework for gathering goal-based requirements," http://eprints.eemcs.utwente.nl/9615/, Enschede, Technical Report TR-CTIT-06-75, December 2006.