David Chadwick, Ilsun You and Hang Bae Chang (Eds.)

The 1st International Workshop on Managing Insider Security Threats (MIST 2009)

Online Proceedings

Purdue University, West Lafayette, USA

June 15-19, 2009

(In Conjunction with IFIPTM 2009)

Preface

As the use of information technology continues to rapidly expand, so do the opportunities for attacking an organization's digital information. During the past decade, information security has primarily focused on preventing illegal attacks by outsiders. However, statistics reveal that organizations loose more resources from insider attacks than from external ones. Consequently organizations are shifting a greater proportion of their security activities from the reduction of external risks to the reduction of internal risks, whether they be from malicious or simply negligent acts.

The first international workshop on Managing Insider Security Threats (MIST 2009) is aimed at providing a showcase for the latest developments in protecting against insider attacks and mistakes, and a forum for discussing the latest research and best practice, as well as an opportunity for determining where future research is still needed. These proceedings will be of interest to information security officers, security researchers, security consultants and enterprise decision makers with security or risk management responsibilities.

We would like to thank all the authors for their submissions, our Program Committee for performing their detailed reviews and feedback to the authors, and our Organizing Committee for their assistance in preparing for this event.

June 2009

David Chadwick and Ilsun You - General Co-Chairs

Hang Bae Chang - Publicity and Web Chair

Organization

General Co-Chairs

David Chadwick (University of Kent, UK)
Ilsun You (Korean Bible University, South Korea)

Publicity and Web Chair

Hang Bae Chang (Daejin University, South Korea)

Program Committee

Gail-Joon Ahn (Arizona State University, USA)
Matt Bishop (University of California, Davis, USA)
Klemens Bohm (University of Karlsruhe, German)
Dawn M. Cappelli (Carnegie Mellon University, USA)
Fariborz Farahmand (Purdue University, USA)
Carrie Gates (CA Labs, USA)
Arif Ghafoor (Purdue University, USA)
Yong Guan (Iowa State University, USA)
Sushil Jajodia (George Mason University, USA)
Byoung-Soo Koh (DigiCAPS Co., Ltd, South Korea)
Dong Seong Kim (Duke University, USA)
Kwangjo Kim (Information and Communications University, South Korea)
Yang Hoon Kim (Daejin University, South Korea)
Hong Joo Lee (Yonsei University, South Korea)
Chu-Hsing Lin (Tunghai University, Taiwan)
Tom Longstaff (Carnegie Mellon University, USA)
Chris Mitchell (University of London, UK)
Peter G. Neumann (SRI, USA)
Gunther Pernul (University of Regensburg, Germany)
Stelios Sidiroglou-Douskos (MIT, USA)
Eugene Spafford (Purdue University, USA)
Shambhu Upadhyaya (SUNY Buffalo, USA)
Michael Wellman (University of Michigan, USA)
Seong-Moo Yoo (University of Alabama In Huntsville, USA)
Meng Yu (Western Illinois University, USA)

Table of Contents

Insider Theft of Intellectual Property for Business Advantage: A Preliminary Model

Andrew P. Moore apm@cert.org, Dawn M. Cappelli dmc@cert.org,
Thomas C. Caron[1] tcaron@cert.org, Eric Shaw[2] eshaw@msn.com,
Randall F. Trzeciak rft@cert.org

CERT®[3] Program, Software Engineering Institute and
CyLab at Carnegie Mellon University
4555 Fifth Avenue
Pittsburgh, PA 15213

Abstract. A study conducted by the Carnegie Mellon University Software Engineering Institute CERT Program analyzed hundreds of insider cyber crimes across U.S. critical infrastructure sectors. Follow-up work involved detailed group modeling and analysis of 35 cases of insider theft of intellectual property. In the context of this paper, insider theft of intellectual property for business advantage includes incidents in which the insider's primary goal is stealing confidential or proprietary information from the organization with the intent to use it to take to a new job, to get a new job, or to start a business. It does not include cases of in which insiders sell an organization's information. This paper describes general observations about, and a system dynamics model of, this class of insider crime based on our empirical data. This work generates empirically-based hypotheses for validation and a basis for identifying mititgative measures in future work.

1 Introduction

Since 2002, the CERT Program at Carnegie Mellon University's Software Engineering Institute has been gathering and analyzing actual malicious insider incidents, including IT sabotage, fraud, theft of confidential or proprietary information, espionage, and potential threats to the critical infrastructure of the United States.[4] Consequences of malicious insider incidents include financial losses,

---

[1] Tom Caron is also a student at the H. John Heinz III College, School of Information Systems Management, Carnegie Mellon University.

[2] Dr. Eric Shaw is a Visiting Scientist at CERT and clinical psychologist at Consulting & Clinical Psychology, Ltd.

[3] CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

[4] "Insiders" include current and former employees, contractors, or other business partners who have or had authorized access to their organization's systems, data, and networks. Insiders

operational impacts, damage to reputation, and harm to individuals. The actions of a single insider have caused damage to organizations ranging from a few lost staff hours to negative publicity and financial damage so extensive that businesses have been forced to lay off employees and even close operations. Furthermore, insider incidents can have repercussions beyond the affected organization, disrupting operations or services critical to a specific sector, or creating serious risks to public safety and national security.

Many models exist to help understand computer-related malicious insider activity, including

- The Capability, Motive, Opportunity Model (Parker, 1998) (Wood, 2002)
- Behavioural models (Suler, 1997) (Shaw, Ruby, & Post, 1998)
- An entity relationship model in a comprehensive characterization framework[5] (Spafford, 2002)
- A criminological and social model (Gudaitis, 1998)

The Defense Personnel Security Research Center (PERSEREC) has produced a vast amount of invaluable data over the years on both espionage and insider threat generally. (Fischer, 2003) (Herbig & Wiskoff, 2002) In one article, a multiple case study approach was used to examine 10 cases of malicious insider IT activity in critical infrastructures drawn from the population of PERSEREC cases. (Shaw & Fischer, 2005) In addition, the Institute for Information Infrastructure Protection (I3P) has brought a wide range of researchers in industry and government to bear on the insider threat problem. [6]

CERT's insider threat work, referred to as MERIT (Management and Education of the Risk of Insider Threat), utilizes the wealth of empirical data collected by CERT to provide an overview of the complexity of insider events for organization—especially the unintended consequences of policies, practices, technology, efforts to manage insider risk, and organizational culture over time.[7] As part of MERIT, we have been using system dynamics modelling and simulation to better understand and communicate the threat to an organization's information technology (IT) systems posed by malicious current or former employees or contractors. Our work began with a collaborative group modeling workshop on insider threat hosted by CERT and facilitated by members of what has evolved into the Security Dynamics Network and the Security Special Interest Group (Anderson, et al., July 2004).

Based on our initial modeling work and our analysis of cases, we have found that different classes of insider crimes exhibit different patterns of problematic behavior and miitigative measures. CERT has found four broad types of insider threat cases

---

are familiar with internal policies, procedures, and technology and can exploit that knowledge to facilitate attacks and even collude with external attackers.

[5] Unpublished manuscript: Tuglular and Spafford, "A Framework for Characterization of Insider Computer Misuse.

[6] See http://www.thei3p.org/research/insider_threat.html.

[7] CERT's insider threat research is published on http://www.cert.org/insider_threat. Early research was funded by the U.S. Secret Service and the Department of Homeland Security, Office of Science and Technology. Our current work including MERIT was funded by Carnegie Mellon University CyLab.

based on the patterns that we have seen in cases identified: IT sabotage, theft or modification of information for financial gain (fraud), theft of intellectual property (IP) for business advantage, and national security espionage. In this paper, we focus on theft of IP for Business Advantage. Our past work has involved modeling insider fraud (Rich, et al., July 2005), insider IT sabotage (Moore, Cappelli, & Trzeciak, 2008)(Cappelli, Desai, Moore, Shimeall, Weaver, & Willke, July 2006), and espionage (Band, Cappelli, Fischer, Moore, Shaw, & Trzeciak, December 2006).

This paper describes our most recent efforts to model aspects of the insider threat problem. We define insider theft of intellectual property for business advantage as crimes in which current or former employees, contractors, or business partners intentionally exceeded or misused an authorized level of access to networks, systems or data to steal confidential or proprietary information from the organization and use it getting another job, helping a new employer or promoting their own side business. Cases where the insider was primarily motivated by personal financial gain have significantly different patterns of behavior and have been excluded from this study (Cappelli, Moore, Trzeciak, & Shimeall, September 2008). While an argument can be made that theft of confidential or proprietary information may ultimately be about money, insiders in this class of cases generally had longer term ambitions, such as stealing the information to get a new job, to succeed in a new job with a competing business, to start a competing business, or to give the stolen data to a foreign government or organization.

This paper is centered on two dominant scenarios found within the cases - the Entitled Independent Scenario and the Ambitious Leader Scenario. We first define our approach to building these models. Next we incrementally build the models describing them as we go. Finally we finish up with general observations and future work. Appendix A summarizes important characteristics of the crimes involving theft of IP for business advantage. Appendices B and C provide an overview of the models developed. We believe that these models will help people understand the complex nature of this class of threat better. Through improved understanding comes better awareness and intuition regarding the effectiveness of countermeasures against the crime. Our work generates strong hypotheses based on empirical evidence. Future work will involve alignment with existing theory, testing of these hypotheses based on random sampling from larger populations, and analysis of mitigation approaches.

2 Approach

Our research approach is based on the comparative case study methodology (Yin, 2003). Cases selected were those fitting the above definition of Theft of IP for business advantage. Cases were identified through public reporting and included primary source materials, such as court records in criminal justice databases (found through searches on Lexis court databases), and other secondary source materials such as media reports (found through searches on Lexis-Nexis news databases and Internet search engines such as Google).

The following criteria are used for case selection:

- The crime occurred in the United States.
- The subject of the crime was prosecuted in a United States Court.
- Sufficient quantities and quality of data was available to understand the nature of the case.

We identified and analyzed 35 cases of theft of intellectual property that satisfied these criteria. The findings from case study comparisons in general, and our study in particular, cannot be generalized with any degree of confidence to a larger universe of cases of the same class or category. What this method can provide, however, is an understanding of the contextual factors that surround and influence the event.

The sole purpose of our modeling effort is precisely that – to help people understand the complex nature of the threat better. Our models evolved through a series of group data analysis sessions with individuals experienced on both the behavioral and technical aspects of insider crimes. We used the system dynamics approach - a method for modeling and analyzing the holistic behavior of complex problems as they evolve over time. [8] System dynamics provides particularly useful insight into difficult management situations in which the best efforts to solve a problem actually make it worse. System dynamics model boundaries are drawn so that all the variables necessary to generate and understand problematic behavior are contained within them. This approach encourages the inclusion of soft (as well as hard) factors in the model, such as policy-related, procedural, administrator, or cultural factors. In system dynamics models, arrows represent the pair-wise influence of the variable at the source of the arrow on the variable at the target of the arrow. Basically, a solid arrow indicates that the values of the variables move in the same direction, whereas a dashed arrow indicates that they move in the opposite direction.

A powerful tenet of system dynamics is that the dynamic complexity of problematic behavior is captured by the underlying feedback structure of that behavior. System dynamics models identify two types of feedback loops: balancing and reinforcing. Significant feedback loops are indicated in the model using a loop label appearing in parentheses in the middle of the loop. Reinforcing loops - indicated by a label with a R followed by a number - describe system aspects that tend to drive variable values consistently upward or downward and are often typified by escalating problematic behaviors. Balancing loops - indicated by a label with a B followed by a number – tend to drive variables to some goal state and are often typified by aspect that control problematic behaviors. For those with color copies of the paper, loops are additionally distinguished by color, where blue arrows are not part of a significant feedback loop.

3 The Entitled Independent Model

This section describes the system dynamics model of the Entitled Independent, an ambitious insider acting alone to steal information to take to a new job or to his own

[8] For more information about system dynamics refer to http://www.systemdynamics.org/.

side business. Note that in most cases the insider had no specific plans to use the information (80%).

3.1 Entitlement

The degree to which insiders felt entitled to information that they stole is difficult to quantify without group interview data. However, feedback from a small sample of subjects, along with the finding that many insiders stole information from their project area, despite having signed intellectual property agreements, support this observation. Almost all of the Entitled Independents stole information in their area of responsibility and about half were at least partially involved with the development of the information stolen. Just over 44% of the Entitled Independents stole information or products even though they had signed IP agreements with the organization. The strong sense of entitlement is seen in this class of insiders when considering that nearly ¾ of the insiders stole information that they had at least partially developed or for which they had signed an IP agreement.

Figure 1 shows the escalation of entitlement to information developed by the insider. As shown in the upper right hand corner, an employee comes into an organization with a desire to contribute to its efforts. As the insider invests time in developing or creating information or products, his contribution to the organization becomes tangible. These individuals, unlike their coworkers, have personal predispositions[9] which result in a sense of entitlement to the information created by the group (yellow loop). This entitlement is shown in the self-reinforcing loop shown in purple and labeled R1 in the figure.

This sense of feeling entitled can be particularly acute if the insider perceives his role in the development of products as especially important. If the insider's work is focused on the contribution to a particular product, for example a commercial software package, or the development of specific business information like customer contact lists, he may have a great sense of ownership of that product or information, leading to even greater sense of entitlement. This self-reinforcing is shown in yellow and labeled R2. In addition, consistent with good management practice, individuals may receive positive feedback for their efforts which these subjects may interpret as particularly reinforcing, given their predispositions. In a recent insider case, one of the authors encountered a subject at significant insider risk who had been told his efforts had saved the company "millions of dollars." This compliment had the unintended consequence of reinforcing the entitlement loop.

Evidence of entitlement was extreme in a few cases. One Entitled Independent who had stolen and marketed a copy of his employer's critical software created a lengthy

[9] Personal predispositions refer to characteristics of the individual that can contribute to the risk of behaviors leading to insider crimes, as well as to the form of these actions, their continuation, and escalation. Personal predispositions such as entitlement were determined by case review by a clinical psychologist trained in remote assessment using a inventory of observable behaviors derived from the American Psychiatric Association's diagnostic criteria for personality disorders.

manuscript detailing his innocence and declaring that everyone at the trial had lied. After being denied a raise, another insider stole the company's client database and threatened to put them out of business on his way out the door.

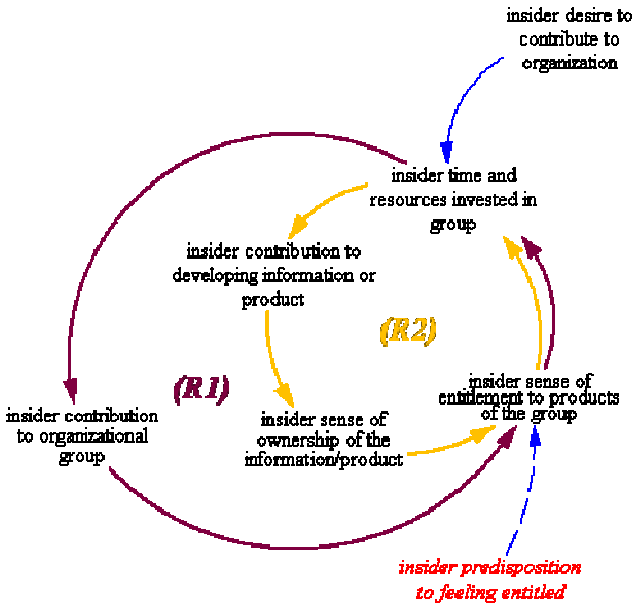


Fig. 1. Insider Entitlement

3.2 Dissatisfaction Leading to Compromise

Expressed dissatisfaction played a role in 39% of the Entitled Independent cases. Dissatisfaction was typically due to denial of some request by the insider as shown in Figure 2. Denied requests in the cases often involved raises and benefits, applications for promotion, and requests for relocation. Other dissatisfaction arose due to the threat of layoffs within the victim organization.

The middle of Figure 2 shows that the organization's denial of a request by the insider leads to the insider's dissatisfaction, which in turn decreases the insider's desire to contribute within the organization. This not only affects the time he invests in contributing to the organization, as it relates to Figure 1, but also the insider's ultimate sense of loyalty to the organization. Dissatisfaction often spurred the insider to look for another job. Once a job offer is received and planning to go to a competing organization commences, the insider's desire to steal information increases. This is spurred on by the insider's dissatisfaction with his current employer in combination with his sense of entitlement to products developed by his group. In a third of the cases (33%) the insider used the information to get a new job or to benefit his new

employer in some way. In almost half of the cases (44%) the insider took the information just in case he ever needed it, with no specific plans in mind. One insider actually broke in after he was terminated to find out whether the organization had made any further progress on the product that he had helped develop while he worked there.

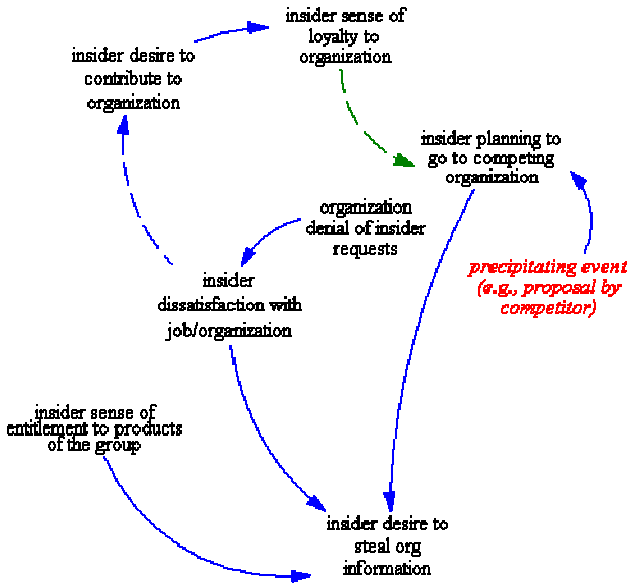


Fig. 2. Insider Dissatisfaction Leading to Compromise

3.3 Theft and Deception

The insider's plan to go to a competing organization, dissatisfaction with his job and/or the organization, combined with the sense of entitlement to the products on which he has been working all contribute to the decision to steal the information. As shown in Figure 3, eventually the desire to steal information becomes strong enough, leading to the theft and its potential exposure to the organization. Exposure includes anything that an organization might observe about the employee's actions or consequences of those actions that indicates heightened risk of insider compromise, whether or not the organization actually makes those observations.

Concern over being caught may make the insider think twice about stealing the information, as shown in the balancing loop labeled B1. Because our data consists of insiders who were caught and prosecuted, we do not know how many subjects may be deterred from insider acts by such concerns. However, our Entitled Independents, did not exhibit great concern with being caught. This lack of concern is consistent with, and may be proportional to, the psychological predispositions that contribute to entitlement. Such individuals tend to overestimate their abilities and underestimate the

capabilities of others. Despite intellectual property agreements being in place in 44% of the cases, less than a quarter of the Entitled Independents explicitly attempted to deceive the organization while taking information.

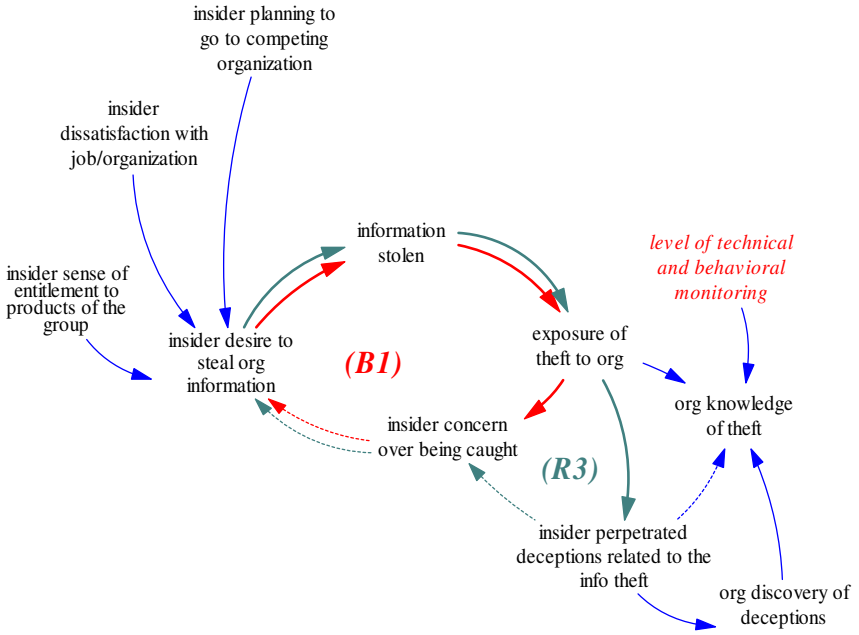


Fig. 3. Insider Theft and Deception

Nevertheless, explicit deception can lessen the insider's concern over being caught, and should be anticipated by a vigilant organization. This is shown in the self-reinforcing loop labeled R3. This loop expresses the intuitive relationship that deception relieves the insider's concern over being caught, thus emboldening his theft of information. The fact that most insiders did *not* often feel it necessary to explicitly deceive the organization regarding the theft is interesting, suggesting the sense of entitlement and its correlates mentioned above, may be particularly strong in these cases.

While explicit deception is not a major factor in this class of crimes, the fact that it does occur needs to be recognized. Upon announcement of resignation, one insider lied about having no follow-on employment to his manager, even though he had told a coworker about his new job at a competitor. As shown in the lower right part of Figure 3, deception may be an indicator of problems to come. Deceptions generally make it harder for the organization to sense the risk of theft and that is why the insider engages in such behavior. But if the organization is vigilant, deceptions may be discovered, alerting the organization to increased risk of insider threat. In general, the organization's accurate understanding of their risk is directly related to its ability to

detect the insider's actions, which with sufficient levels of technical and behavioral monitoring may be discoverable. Over half (56%) of the Entitled Independents stole information within one month of resignation, which gives organizations a window of opportunity for discovering the theft prior to employee termination.

3.4 Summary

Appendix B shows the final model of the Entitled Independent. Based on the patterns observed in the cases, half of the insiders who stole proprietary information felt a sense of entitlement to that information, based on their participation in its development, regardless of whether or not they signed an intellectual property agreement. This sense of entitlement, when viewed in light of an event seen as dissatisfying to the insider, formed the catalyst for the insider to begin looking for other jobs. Insiders then used stolen information to pursue new opportunities. The Entitled Independent is usually fully authorized for access to this information and takes it very close to resignation with very little planning. In addition, the Entitled Independent rarely acts as if they are doing anything wrong, probably partly because they feel perfectly entitled to take the information or product with them to their new job.

4 The Ambitious Leader Model

This section describes the Ambitious Leader model. As noted, these cases involve a leader who recruits insiders to steal information for some larger purpose. The cases can be distinguished according to whether the insider

- had specific plans to develop a competing product or use the information to attract clients away from the victim organization (60%), or
- was working with a competing organization to help his new employer (40%).

It also includes cases in which the insider was partially motivated by a desire to contribute to a foreign government or company (we view this an implicit recruitment of insider help). The rest of this section describes additional aspects of the Ambitious Leader model not exhibited by Entitled Independents. This scenario is more complex than the Entitled Independent scenario, involving more intricate planning and deception, as well as new areas such as attempts to gain increased access and recruitment of other employee's into the leader's scheme.

The starting point for our description is almost exactly the same as the Entitled Independent model described above. The primary difference is that there was little evidence of employee dissatisfaction in the Ambitious Leader class (6%), whereas it played a more significant role with Entitled Independents (39%). Insiders in this scenario were motivated not by dissatisfaction but by an Ambitious Leader promising greater rewards. In one case, the head of the public finance department of a securities firm organized his employees to collect documents to take to a competitor. Over one weekend he then sent a resignation letter for himself and each recruit to the head of

the sales department. The entire group of employees started work with the competitor the following week. In another case an outsider who was operating a fictitious company recruited an employee looking for a new job to send him reams of his current employer's proprietary information by email, postal service, and a commercial carrier.

Except for the dissatisfaction of the Entitled Independent, the initial patterns for Ambitious Leaders are exactly the same. In fact the beginning of the Ambitious Leader model is just the model shown in Appendix B without the "Insider Dissatisfaction with Job/Organization" variable shown in the middle left of the model. Theft took place even though intellectual property agreements were in place for about half (46%) of the Ambitious Leader cases. In at least one case, the insider lied when specifically asked if he had returned all proprietary information and software to the company according to the IP agreement he had signed. He later used the stolen software to develop and market a competing product in a foreign country. Almost all of the insiders in the Ambitious Leader cases stole information or products in their area of job responsibility, with over half of those at least partially involved in developing the information or product stolen. These facts strongly suggest that the insiders felt a sense of entitlement to the information or products that they stole.

4.1 Insider Planning of Theft

The Ambitious Leader cases involved a significantly greater amount of planning than the Entitled Independent cases. By definition the cases involved recruiting of insiders which involves a greater amount of planning almost by necessity. Other forms of planning involved:

- Creating a new business (37%),
- Coordination with a competing organization (37%), and
- Collecting information in advance of the theft (60%).

This aspect of the insider behavior is reflected in the balancing loop labeled B2 in Figure 5. The B2 loop parallels the loop B1 from the Entitled Independent model in Figure 4 but describes an additional dimension: the insider's plans to steal information prior to the actual theft. This potential additional point of exposure of the impending theft apparent in the Ambitious Leader cases includes the extensive planning described above and measures by the insider to hide his actions. Most of the cases involved planning by the insider a month or more before the insider's departure from the organization (84%). In almost half of the cases the actual theft took place a month or more before the insider's departure (43%). One insider planned with a competing organization abroad and transferred documents to the company for almost two years prior to her resignation.

About a third (34%) of the insiders committed explicit deceptions to hide their plans for the theft of intellectual property. The self-reinforcing loop labeled R3 is slightly stronger in this case than for the Entitled Independent. In all but one of these cases, the organization had IP agreements with the insiders explicitly stating the organization's ownership of the stolen information. In fact, there was only one case

where an IP agreement was in place between the organization and the insider but no deceptions were committed by the insider. This provides a working hypothesis regarding the effectiveness of an organization's efforts to promote its concern about IP theft. If the organizations involved publicized their concern and pursued violations, this may have increased the odds of deception while providing another observable indicator of insider risk.

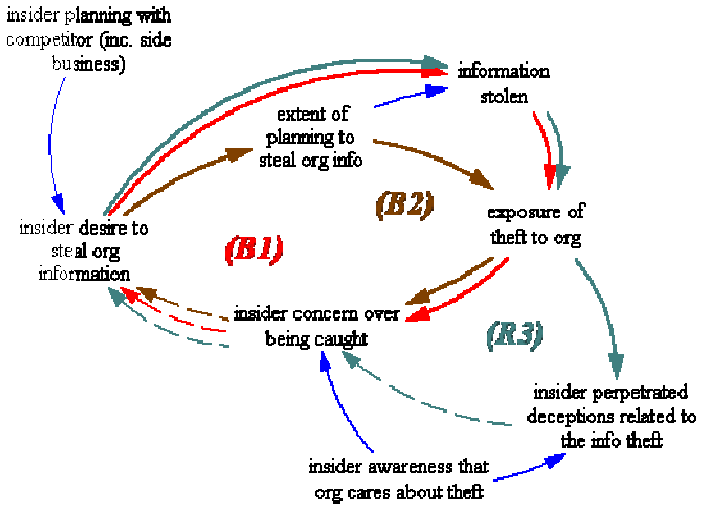


Fig. 4. Theft Planning by Ambitious Leader

4.2 Increasing Access

The amount of planning by the Ambitious Leader and the insiders under his control appears to depend on the extent that any one participant has access to all of the information targeted for theft. The more segregation of privilege the more planning, participation, and coordination needed to commit the theft. In over half (55%) of the Ambitious Leader cases, the primary insider had authorization for only ***part*** of the information targeted and had to take steps to gain additional access. In the previously mentioned case the insider who transferred proprietary documents to a foreign company for almost two years asked her supervisor to work on a special project that would increase her access to highly sensitive information in the weeks prior to her leaving the country with a company laptop and numerous company documents, both physical and electronic. This is in stark contrast to the Entitled Independent cases where two-thirds (67%) of the primary insiders were authorized to access ***all*** of the information stolen.

As shown on the right side of Figure 6, a primary means of extending the access of the Ambitious Leader to more information is the recruiting of insiders. The recruitment of insiders increases the amount of planning activity necessary to

coordinate insider activities. As shown in the self-reinforcing loop labeled R4 in the figure, as the insider invests more time and resources into the plans for theft and movement to the competing organization, it is less and less likely that they will back out of those plans.

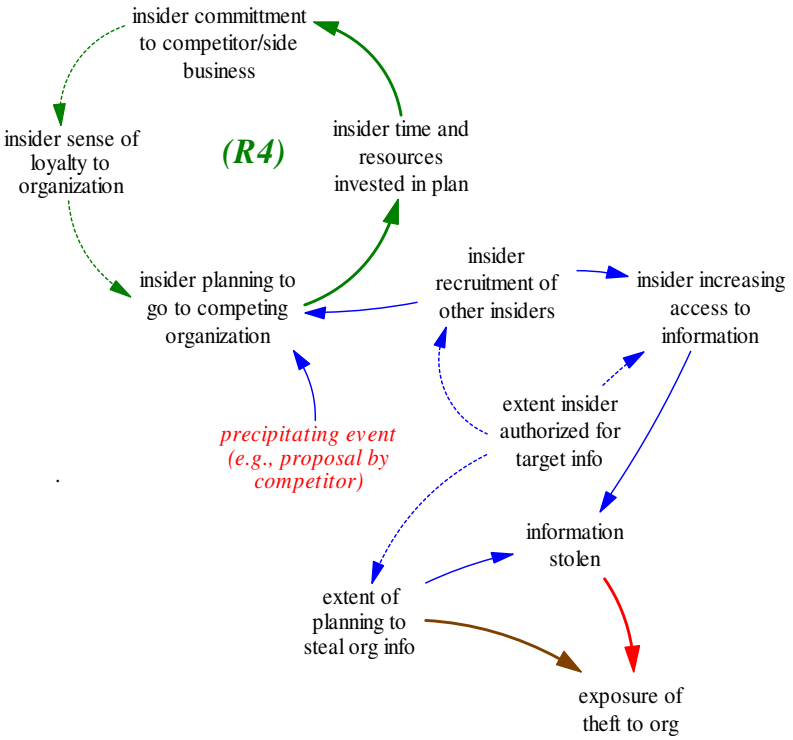


Fig. 5. Increasing Access by the Ambitious Leader

While we can't know for sure that the R4 loop's self-reinforcement of insider criminal behavior is what is happening in these cases, there is strong evidence in the psychological literature for the "sunk cost effect." (Sastry, 1998) The sunk cost effect involves an irreversible investment (e.g., time spent planning a theft) that decision-makers consider as powerful motivation to continue the action. The further investment is justified not in terms of the initial rationale but because so much has already been invested (Staw & Ross, 1989).

There is evidence of this self-reinforcing pattern in one case of a job-hunting insider who met someone online claiming falsely to own a competing business. While at first reluctant to send proprietary information, as the "friendship" grew and requests for confidential information repeated, the insider gradually sent more and more of her employer's confidential information to the outsider seemingly unable to stop herself. This indicates that insiders may be reluctant to back out of the plans because others

are depending on them to carry out their part of the crime, not the least of which is the Ambitious Leader. The social costs of withdrawal from the scheme may be too high, thus further motivating insiders to continue their involvement, even if they know it is wrong and would like to back out.

4.3 Organization Knowledge of Theft

There are many more avenues for an organization to become aware of heightened risk of insider theft of IP in the Ambitious Leader cases than in the Entitled Independent cases. The Entitled Independent is usually fully authorized for access to the information taken and takes the data very close to resignation with very little planning. In addition, the Entitled Independent rarely acts as if they are doing anything wrong, probably partly because they feel a proprietary attachment to the information or product. The Ambitious Leader, on the other hand, often has to gain access to information for which he is not authorized. This involves, in part, coordinating the activities of other insiders and committing deceptions to cover up the extensive planning that generally takes place.

Figure 7 illustrates the avenues available for an organization to continually assess the risk they face regarding theft of intellectual property. At the bottom of the figure, the discovery of insider deceptions may even be a better means to detect heightened insider risk here than in the Entitled Independent cases due to their greater prominence in these cases. In some of the cases that we reviewed, the organization found out about the theft because the insider tried to use the information. Two primary uses were observed: marketing of the competing product to the general public or to the victim organization's customers, and soliciting the business of the victim organization's customers. While these two uses are not extremely different they do differ based on what was stolen – in the first case, the organization's product (e.g., software system) and in the second case client information (e.g., organization business plans or client points of contact). In one case the insider had stolen source code for a product being marketed by his previous employer and was demonstrating a slightly modified version at a trade show. Unfortunately for him, his previous co-workers observed the activity and alerted the authorities. While this detection is later than one would prefer, it is still not too late to take action and prevent further losses.

Earlier detection of plans to steal or actual theft by an insider may occur through technical monitoring of systems. Over half (56%) of the Entitled Independents and almost two-thirds (67%) of the Ambitious Leader insiders stole information within one month of resignation. Many of these involved large downloads of information outside the patterns of normal behavior by those employees. In over one-third (38%) of the cases of Ambitious Leaders, an insider emailed or otherwise electronically transmitted information or plans from an organizational computer. Keeping track of backup tapes is also important – in the case described in the previous paragraph, the insider took the backup tape from his computer on his last day of work. Understanding the potential relevance of these types of precursors provides a window of opportunity for organizations to detect theft prior to employee termination.

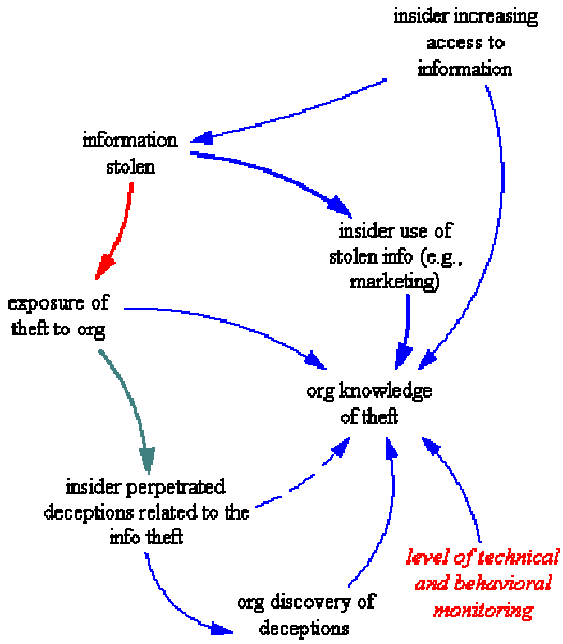


Fig. 6. Organization Knowledge of Theft of IP in Ambitious Leader Cases

Of course, the earlier an organization can become aware of such plans the better. This depends on behavioral as well as technical monitoring and is more likely to catch incidents involving Ambitious Leaders than Entitled Independents. Here the organization needs to look for evolving plans and collusion by insiders to steal information, including attempts to gain access to information over and above what an employee is authorized in over 2/3 (69%) of the cases. One insider, over a period of several years, exhibited suspicious patterns of foreign travel and remote access to organizational systems while claiming medical sick leave. It is not always this blatant but signs are often observable if an organization is vigilant.

4.4 Insider IP Theft Benefiting a Foreign Entity

Nine of the 35 cases (26%) of theft of intellectual property were intended to benefit a foreign government or company. All of these cases fit the model of the Ambitious Leader Scenario and were included in the statistics reported in this section. In these cases, loyalty to their native country trumped loyalty to the employer. Similar to the way insiders in the other cases were motivated by an Ambitious Leader, insiders with an affinity toward a foreign country were motivated by the goal of bringing value to, and sometimes eventually relocating in, that country. In all of the Ambitious Leader cases, there is an influencing individual and motive acting on the subject to promote the criminal act.

4.5 Summary

While half of the cases involved insiders acting as Entitled Individuals, the other half were characterized by Ambitious Leaders acting as the insider or guiding the insider to steal information. The final model of the Ambitious Leader is shown in Appendix C. Ambitious Leader cases involved much more planning and deception, as insiders typically did not initially have access to the data in question. These attacks were more likely to occur closer to the point at which the insider left the organization. In some cases, the ambitious leader was an agent of a foreign interest, and the theft of information was geared toward the benefit of a foreign entity.

5 Conclusion

This paper describes two models of insider theft of intellectual property for business advantage developed using empirical data from cases involving actual insider compromise. The following key observations describe the overarching patterns in the cases of insider theft of intellectual property.

- Many insiders exhibited a sense of entitlement to the information they stole. Insiders generally disregarded IP agreements (44%).
- Many Entitled Independents showed signs of dissatisfaction with some aspect of their job, often compensation, benefits, or promotions (39%). No insiders stealing for the benefit of a foreign government or company showed signs of dissatisfaction.
- The insiders were evenly split according to whether they had authorized access to only part or whether they had authorized access to all of the information stolen. The majority of Entitled Independents had authorized access to the information they stole (67%). The majority of Ambitious Leaders did not have authorized access to all of the information they stole (69%).
- Most insiders were involved with significant planning activities more than a month before resignation. (59%).
- Some insiders started stealing information more than 1 month prior to their departure. (21%).
- Most insiders stole at least some information within a month of resignation (65%).
- Most insiders stole information in their area of job responsibility (74%) and many at least partially developed the information/product stolen (41%).

This work has focused on gaining a more rigorous understanding of the nature of the threat and providing an effective means for communicating that to the general public. We have found that the system dynamics approach helped to structure and focus the team's discussion. This was particularly important since members of the team, by necessity, came from the different disciplines of psychology and information security. The models also provided a concrete target for validation through mapping to observables exhibited by the real-world cases.

Of course, this is only the beginning of the work. Future work needs to further validate the hypotheses embodied in the model. In addition, our ultimate concern is to develop effective measures to counter the problem of theft of intellectual property. Significant methodological and data challenges must be overcome before research on insider activity can be soundly prescriptive for mitigation policies, practices, and technology. However, we cannot overestimate the importance of looking at the total context of adverse insider behavior for understanding why these events happened and how they might be prevented in the future.

By using the system dynamics approach we will attempt to assess the weight and interrelatedness of personal, organizational, social, and technical factors. We expect future work to use modeling and simulation to identify and evaluate the effectiveness of deterrent measures in the workplace. Prospective studies of these phenomena will always be challenging because of low base rates. In the meantime, system dynamics modeling using available empirical data can bridge this methodological gap and translate the best available data into implications for policies, practices, and technologies to mitigate insider threat.

6 Acknowledgements

CERT would like to thank the Army Research Office and Carnegie Mellon University's CyLab for funding this project. Our original insider threat work was funded by the U.S. Secret Service whose support we will always be grateful for. We would also like to thank the following for their contributions to our insider threat efforts and this project: Daniel Phelps of the CERT, Christopher Nguyen and Hannah Joseph - prior CyLab employees and graduates of the Information Networking Institute of Carnegie Mellon University – Michael Hanley and Greg Longo –current CERT employees and students of the Heinz College of Carnegie Mellon University.

7 Bibliography

Anderson, D. F., Cappelli, D. M., Gonzalez, J. J., Mojahedzadeh, M., Moore, A. P., Rich, E., et al. (July 2004). Preliminary System Dynamics Maps of the Insider Cyber-Threat Problem. *Proceedings of the 22nd International Conference of the System Dynamics Society.*

Band, S. R., Cappelli, D. M., Fischer, L. F., Moore, A. P., Shaw, E. D., & Trzeciak, R. F. (December 2006). *Comparing Insider IT Sabotage and Espionage: A Model-Based Analysis.* Carnegie Mellon University, Software Engineering Institute.

Cappelli, D. M., Desai, A. G., Moore, A. P., Shimeall, T. J., Weaver, E. A., & Willke, B. J. (July 2006). Management and Education of the Risk of Insider Threat (MERIT): Mitigating the Risk of Sabotage to Employers' Information, Systems, or Networks. *Proceedings of the 24th International System Dynamics Conference.* Nijmegen, Netherlands.

Cappelli, D. M., Moore, A. P., Trzeciak, R. F., & Shimeall, T. J. (September 2008). *Common Sense Guide to Prevention and Detection of Insider Threat* (3rd ed.). CERT Program, Software Engineering Institute, and CyLab of Carnegie Mellon.

Fischer, L. (2003). *Characterizing Information Systems Insider Offenders.* Pensacola, FL: International Military Testing Association Proceedings.

Gudaitis, T. (1998). *The missing link in information security: Three Dimensional Profiling"* (Vol. 1). Cyber Psychology and Behavior.

Herbig, K. L., & Wiskoff, M. (2002). *Espionage Against the United States by American Citizens 1947-2001.* Defense Personnel Security Research Center.

Meadows, D. L., Behrens, W. W., Meadows, D. H., Naill, R. F., Randers, J., & Zahn, E. K. (1974). *Dynamics of Growth in a Finite World.* Cambridge, MA: Wright Allen Press, Inc.

Moore, A. P., Cappelli, D. M., & Trzeciak, R. F. (2008). *The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures* (Vol. Insider Attack and Cyber Secruity: Beyond the Hacker). (S. Stolfo, S. M. Bellovin, S. Hershkop, A. Keromytis, S. Sinclair, & S. W. Smith, Eds.) New York, NY: Springer Science+Business Media, LLC.

Parker, D. B. (1998). *Fighting Computer Crime: A New Framework for Protecting Information.* New York: John Wiley and Sons.

Rich, E., Martinez-Moyano, I. J., Conrad, S., Cappelli, D. M., Moore, A. P., Shimeall, T. J., et al. (July 2005). Simulating Insider Cyber-Threat Risks: A Model-Based Case and a Case-Based Model. *Proceedings of the 16th International Conference of the System Dynamics Society.* Quebec City, Canada.

Sastry, M. A. (1998). Analyzing the research on self reinforcing processes in organization: Another approach to archetypes. *Proceedings of the 16th International Conference of the System Dynamics Society.* Quebec City, Canada.

Shaw, E., & Fischer, L. G. (2005). *Ten Tales of Betrayal: The Threat to Corporate Infrastructure by Information Technology Insiders.* Defense Technical Information Center.

Shaw, E., Ruby, K. G., & Post, J. M. (1998). *The Insider Threat to Information Systems: The Psychology of the Dangerous Insider* (Vols. 2-98). Security Awareness Bulletin.

Spafford, E. (2002). *A Framework for Understanding and Predicting Insider Attacks* (Vol. COMPSEC 2002). London: Elsevier Science Ltd.

Staw, B. M., & Ross, J. (1989). Understanding Behavior in Escalation Situations. *Science , 246*, 216-220.

Sterman, J. D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World.* New York, NY: McGraw-Hill.

Suler, J. (1997). *The Bad Boys of Cyberspace: Deviant Behaviour in On-Line Multimedia Communities and Strategies for Managing It.* http://www.rider.edu/~suler/psycyber/badboys.html.

Wood, B. (2002). *An Insider Threat Model for Adversary Simulation.* RAND.

Yin, R. K. (2003). *Case Study Research.* (3rd, Ed.) Thousand Oaks: Sage Publications.

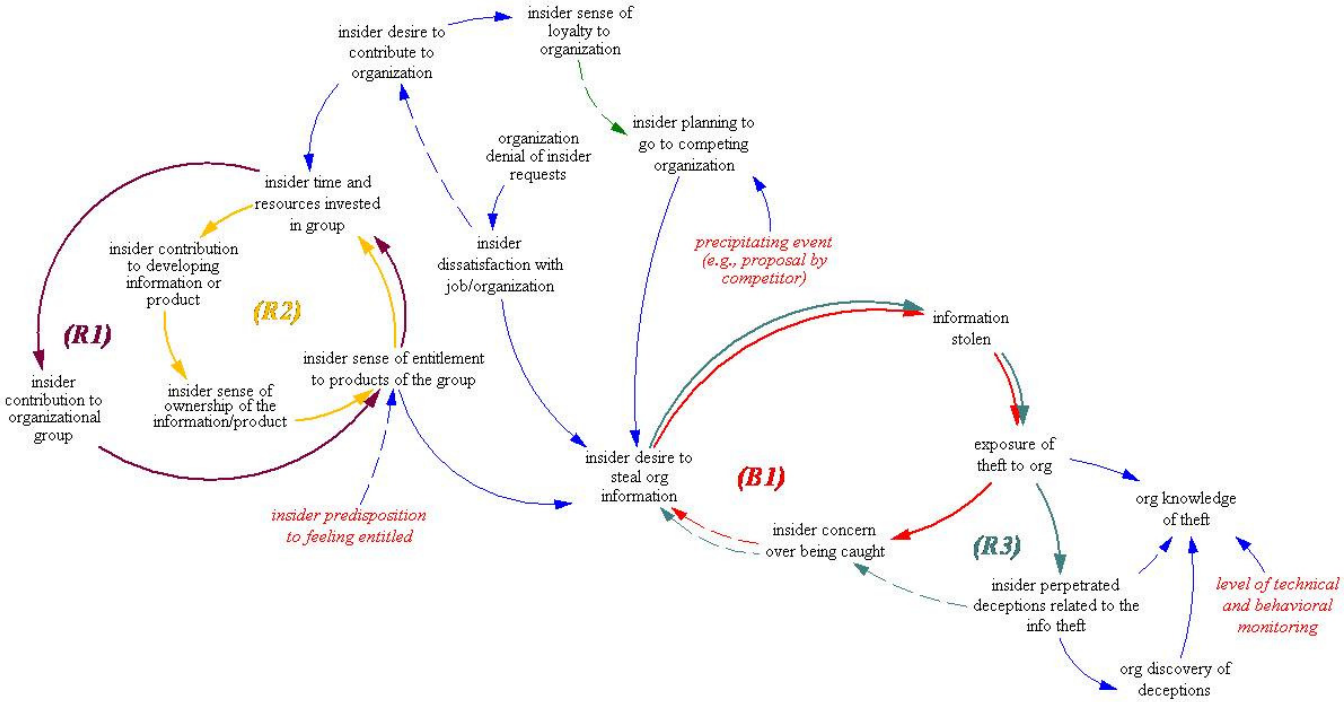
Appendix A: Nature of Insider IP Theft for Business Advantage

Who were the insiders?	• 91% of the insiders who stole intellectual property were male (males comprise 82% of CERT's overall case repository where gender is known). • 55% held technical positions (technical positions comprised 56% of the overall case repository where positions were known). • 65% were current employees when they committed their illicit activity (current employees comprise 70% of CERT's case repository where employment status is known). • Nearly 80% of the insiders had already accepted positions with another company or had started a competing company at the time of the theft.
Why did they do it?	• 32 % of the insiders stole the information to gain an immediate advantage at a new job. • In 21% of the cases, the insider gave the information to a foreign company or government organization. The average financial impact for cases involving the benefit for a foreign entity was over four times that of domestic intellectual property theft.
When did the attacks happen?	• 73% of the crimes where information was available were committed during working hours (37% of CERT's overall cases were committed during work hours). • 37% stole within a month of their departure from the organization (this characteristic drops to 7% when viewed across all crimes in the CERT repository). • Less than one third of the insiders continued their theft for more than one month; and of those that did so, half of them stole the information for a side business, and half to take to a new employer.
How did they attack?	• Over three-quarters of the insiders had authorized access to the information stolen at the time of the theft. (27% of the insiders across all crimes had authorized access at the time of the theft). • None of the insiders had privileged access[10], which enabled them to commit the crime (6% of all crimes involved an insider with privileged access). • In approximately 15% of the cases, the insider colluded with at least one other insider to commit the crime (insiders collaborated with accomplices 22% of the time overall). • The insider was only actively recruited by someone outside the organization in less than 25% of the cases.

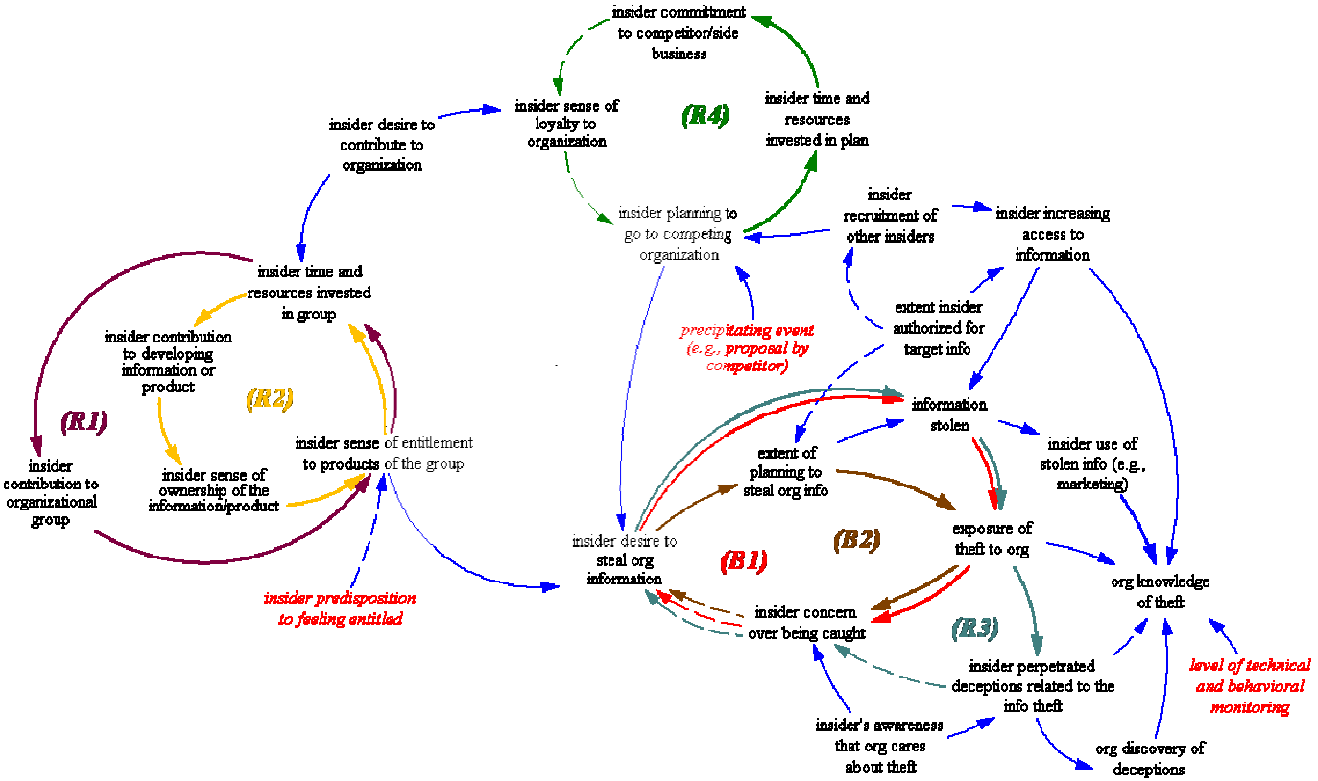
[10] Such as that given to a system or database administrator.

	• 68% of the insider attacked at the workplace (21% attacked remotely, accessing their employers' networks from their homes or from another organization. In 11% of the cases the location of the attack was unknown.)
How was the theft detected?	• Many of these incidents were detected by non-technical means, such as: o notification by a customer or other informant, o detection by law enforcement investigating the reports of the theft by victims, o reporting of suspicious activity by co-workers, and o sudden emergence of new competing organizations. • The most likely person to discover an insider theft for business advantage is a non-technical employee. In cases where we were able to isolate the person who discovered the incident, 57% were detected by non-technical employees (non-technical employees were responsible for discovering insider crime in 36% of the overall case repository).
What were the impacts?	• In 26% of the cases, proprietary software or source code was stolen (insiders targeted software in 8% of the entire CERT case repository). • 29% of cases involved business plans, proposals, and other strategic plans (insiders targeted business plans in 5% of the entire CERT case repository). • 63% involved trade secrets, such as product designs or formulas (trade secrets were stolen in 15% of the cases in CERT's repository, regardless of crime type). • 20% involved customer lists or customer data (This information was targeted 23% of the time across all crimes). • 20% involved the organization's physical property (physical property was the target in 8% of CERT's cases overall).

Appendix A: Entitled Independent Model of the Insider IP Theft



Appendix B: Ambitious Leader Model of the insider IP Theft



Insider Behavior: An Analysis of Decision under Risk

Fariborz Farahmand and Eugene H. Spafford

Center for Education and Research in Information Assurance and Security
Purdue University, West Lafayette, Indiana, USA
{fariborz,spaf}@purdue.edu *

Abstract. There is considerable research being conducted on insider threats is directed to developing new technologies. At the same time, existing technology is not being fully utilized because of non-technological issues that pertain to economics and the human dimension. Issues related to how insiders actually behave are critical to ensuring that the best technologies are meeting their intended purpose. In our research, we have investigated accepted models of perceptions of risk and characteristics unique to insider threat, and we have introduced ordinal scales to these models to measure insider perceptions of risk. We have also investigated decision theories, leading to a conclusion that Prospect Theory, developed by Tversky and Kahneman, may be used to describe the risk-taking behavior of insiders and can be accommodated in our model. We discuss the results of validating that model with thirty-five senior information security executives from a variety of organizations. We also discuss how the model may be used to identify characteristics of insiders' perceptions of risk and benefit, their risk-taking behavior and how to frame insider decisions.

1 Who is an Insider?

A survey of the literature identifies several attempts to understand the insider threat and the behavior of insiders in organizations (e.g., [1]; [2]), and to provide technical defense against those threats (e.g., [3]; [4]). Bishop and Gates [5] explain that defining 'insider' as a binary condition is not appropriate and they instead define insiders based on their access attributes. However, currently there is no generally-accepted definition of an insider.

From an organizational perspective, is employment the defining factor? For example, are hours worked per week, or the person's history with that organization the defining aspects? Organizational behavioral studies do not support a mapping between these factors and the extent to which an individual employee perceives self as an insider within a particular organization (e.g., [6]). The main goals of our ongoing research on insider threats are to understand insider risk

* D. Chadwick, I. You and H. Chang (Eds.): Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST2009), Purdue University, West Lafayette, USA, June 16, 2009.

taking behavior and to frame insider decisions on taking actions such as theft of information, sabotage, and fraud in organizations.

2 Insider Perception of Information Security Risks

Fischhoff et al. [7] investigated perceptions of risk, and particularly ways to determine when a product is acceptably safe. Their model can be adopted and used to define insider risk associated with misbehavior:

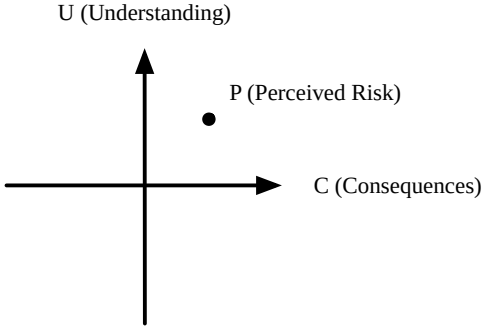
1. Does the insider voluntarily get involved in the risk situation (voluntariness)?
2. To what extent is the risk of consequence from the insider's action to him/her immediate (immediacy of effect)?
3. To what extent are the risks known (precisely) by the insider who is exposed to those risks (knowledge about risk)?
4. To what extent are the risks precisely known and quantified (knowledge to science)?
5. To what extent can the insider, by personal skill or diligence, avoid the consequences to him/her while engaging in the untoward activity (control over risk)?
6. Does the risk affect the insider over time or is it a risk that affects a larger number of people at once (chronic-catastrophic)?
7. Are these risks new to the insider or is there some prior experience/conditioning (newness)?
8. Is this a risk that the insider has rationalized and can think about reasonably calmly (common-dread)?
9. When the risk from the activity is realized in the form of consequences to the insider (severity of consequences)?

It has been shown that unknown risk and dread risk can be used to account for about 80 percent of the results generated by using all nine variables that were originally introduced by Fischhoff and his colleagues (e.g., [8]). (We note that the nine risk factors given above may not apply in extreme cases involving drugs or ideology.)

We formulated a model based on the psychometric model of risk perception developed by Fischhoff, Slovic and others, in which characteristics of a risk are correlated with its acceptance. We then modified that model to accommodate factors present in insider misuse and to condense Fischhoff's nine variables of risk – listed above – by considering understanding (familiarity and experience) and consequences (scope, duration, and impact) to the insider as the two principal characteristics of information security and privacy risks.

If we explore the fear insiders have of the potential effects to them of the risks of perpetrating IT misuse, we can model the consequences of the breach to the insider. To model this, we consider three main questions: 1) How serious are effects perceived by insiders? 2) How immediate are effects on insiders, and 3) How much do insiders fear the effects? Analyzing these questions enables us to assign a simple metric to this dimension of the model. We define five levels of consequence:

Fig. 1. Characteristics of Perceptions of Information Security Risks



1. Level 1: Effects are trivial, temporary and commonplace
2. Level 2: Effects are potentiality serious but treatable/recoverable
3. Level 3: Effects are serious, long term but considered normal
4. Level 4: Effects are serious, ongoing and raise deep concerns
5. Level 5: Effects are catastrophic, ongoing and highly feared

The level definitions ('trivial,' 'serious,' etc.) are based on those published by the National Institute of Standards and Technology (see [9]). Level 5 and level 1 represent the highest and lowest level of consequences to insiders, respectively.

For the second dimension, understanding, we can explore the factors motivating users to consider certain risks while dismissing others. These questions are intended to identify affective factors that influence users' cognitive understanding of cause and effect. This resolves into two main questions: 1) who (among the insider group) understand the hazard? 2) What do insiders know?

Our framework for categorizing understanding is based on the work of Bloom and Krathwhol [10]. In this, our interest is in understanding risk causes and effects using the cognitive domain, and what adds to insiders' motivation to increase understanding using the affective domain. We obtain the following six-level metric for the understanding dimension of our model by answering these questions:

1. Level 1: Evaluation: Can the insider make judgments about the value of ideas or materials?
2. Level 2: Synthesis: Can the insider build a structure or pattern from diverse elements?
3. Level 3: Analysis: How insiders distinguish between facts and inferences.
4. Level 4: Application: How insiders use a concept in a new situation or unprompted use of an abstraction.
5. Level 5: Comprehension: Can the insider understand the problem, for e.g. state a problem in his/her own words?
6. Level 6: Knowledge: Can the insider recall data or information?

Level 6 and level 1 represent the lowest and the highest level of understanding, respectively.

The perceived risk in our model is a function of consequence and understanding. An approximate perceived risk score may be constructed from the consequence metric and the inverse of the understanding metric. The perceived risk score therefore increases whenever the consequences are more severe for insiders, and decreases as the insider gains deeper understanding of the nature and limits of the risk. Some cases may not match this model exactly but this score is nonetheless a good match for many case studies and the experiences of the experts interviewed in our validation study.

If managers understand the dynamic processes by which insiders learn about risk, they can then use that knowledge to choose among alternatives that have different uncertainties, risks and benefits. Our research addresses the dynamics of perception by including a variable time element in our model that causes the risk score to decay with time. That extension will not be discussed here (for full details of this model see [11]) but may be employed as part of a more extensive evaluation of risk perception.

3 Model Validation

To validate our model, we presented it to thirty-five senior information security executives in industry and governmental organizations across the U.S. Following a ten-minute description of our model, we conducted our studies in structured one-on-one meetings and telephone interviews.

During the meetings/interviews we asked these executives if they were able to map the perceived risk of the worst information security incident that they had experienced into our model. We also asked questions such as: Were those incidents caused by insiders or outsiders? How do you describe the level of the consequences and understanding of risks of those incidents? Do you believe this level was the same for all the stakeholders?

These executives each had at least a decade of experience with a large range of information security issues. All these executives were able to map their perceived risk into our model. They were also able to estimate the range of perceived risk by different stakeholders. However, the interviewees stated that perceived risk is not the only factor that we should investigate in modeling insider risk and framing insider decisions, and the perceived benefit is likely to play a more important role in insider decisions.

4 Fraud Triangle

Most of the law enforcement agents who were interviewed in our research indicated the Fraud Triangle was a model that they regularly used when investigating insider crime. Joseph T. Wells [12], a retired law enforcement agent, developed this model as a model of elements supporting and motivating fraud. Mr. Wells's model was influenced by the research of Donald R. Cressey (1919 – 1987), a

sociologist known for his work in organized crime investigation. Motive, opportunity, and rationalization are the three elements of Wells's model, also known as the Fraud Triangle.

Combining our model for risk perception with Wells's model indicates that management should ensure that discovered misuse is punished appropriately, and that appropriate audit controls are in place. The combination further suggests that opportunity may be countered by random observation and unpublicized controls, thus introducing additional uncertainty to the perception of risk.

5 Inverse Relationship between Perceived Risk and Benefit

Similar to the arguments made by decision scientists about the role of affect in human decision making (e.g., [13]), we argue that insiders use an affect heuristic to make judgments. That is, representations of events in insiders' minds are tagged to varying degrees with affect. Insiders consult or refer to an affective pool in the process of making judgments. Using an overall and affective impression can be far easier than weighing the pros and cons or retrieving from memory many relevant examples, especially when the required judgment is complex and includes many unknown variables.

The affect heuristic also predicts that using time pressure to reduce the opportunity for analytic deliberation should enhance the inverse relationship between perceived benefits and risks—the higher the perceived benefit, the lower the perceived risk, and vice versa. Finucane et al. [13] showed that the inverse relationship between perceived risks and benefits increased greatly under time pressure as predicted. This is consistent with Zajonc's findings [14] that affect influences judgment directly and is not simply a response to a prior analytic evaluation.

Kahneman and Lovallo [15] explain the concept of inside view–a forecast is generated by focusing on the case at hand, for e.g., by considering the plan and the obstacles to its completion, and outside view–a focus on the statistics of a class of cases similar in respects to the present one. Our findings indicate that insiders are normally biased in favor of the inside view and tend to neglect the statistics of the past. This characteristic makes them capable of two biases–also known as isolation errors ([15]): Their forecasts of future outcome are often anchored on plans and scenarios of success rather than on past results, and are therefore optimistic; their evaluations of single risky prospects neglect the possibilities of pooling risks.

Another explanation for the inverse relationship between perceived risk and benefit by insiders could be that perceived benefits–compared to perceived risks–are simply more evaluable, largely they are conceptualized unidimensionally, and are psychologically represented in terms of a convenient and numerical scale ([16]). Lichtenstein and Slovic [17] also explain that the amount to win can directly translate to an amount to bid–in an insider's case to take different approaches to commit the crime, or to commit or not to commit the crime at

all. Probabilities of winning and losing, presented in probability units, are more difficult to translate into monetary units. This can lead insiders to decisions that are highly correlated with the amount to win but poorly reflect the variations in probabilities and amount to lose.

6 Framing Insider's Decisions

Classical decision theory ([18], [19]) frame the choice people make in terms of four basic elements:

1. A Set of potential actions (A_i) to choose between,
2. A set of events or world states (E_j),
3. A set of consequences obtained (C_{ij}) for each combination of action and event, and
4. A set of probabilities (P_{ij}) for each combination of action and event

According to classical decision theory, the expected value of an action is calculated by weighting its consequences over all events by the probability the event will occur. Classical decision theories neither adequately explain the insider behavior nor do they assist managers in selecting appropriate control measure(s) to prevent/minimize damage or loss caused by insider misuse. For example, a manger might be deciding whether to install misuse detection software in his company's network. Installing or not installing software responds to two actions A_1 and A_2. The expected consequences of either action depend upon whether misuse occurs. Misuse occurring or not occurring corresponds to two events E_1 and E_2. Installing misuse detection software may reduce the consequences (C_{11}) of misuse occurring. As the probability of misuse occurrence increases, use of software seems to be more attractive.

From probability theory, it can be shown that the return to a manager is maximized by selecting the alternative with the greatest expected value. The expected value of an action A_i is calculated by weighting its consequences C_{ik} over all events k, by the probability P_{ik} the event will occur. The expected value of a given action A_i is therefore:

$$EV[A_i] = \sum_k P_{ik} C_{ik} \tag{1}$$

More generally, a manager's preference for a given consequence C_{ik} might be defined by a value function $V(C_{ik})$, which transforms consequences into preference values. The preference values are then weighed using the same equation. The expected value of a given action A_i becomes:

$$EV[A_i] = \sum_k P_{ik} V(C_{ik}) \tag{2}$$

Expected utility theory extended expected value theory to describe how people make certain economic choices ([18]). Subjective utility theory added the

notion that uncertainty about outcomes could be represented with subjective probabilities ([19]) and multi-attribute utility theory ([20]) extended subjective utility theory to the case where the decision maker has multiple objectives.

Traditional methods of engineering risk analysis and expected utility decisions, despite all their differences, share a common core: Both rely on the assumption of complete rationality. However, the results of studies by decision science researchers in the past four decades contrast with the outcomes of these traditional methods, which stem from the work of Daniel Bernoulli and Thomas Bayes in the seventeenth century. Not all decisions are completely rational.

A large literature has been developed showing that the framing of decisions can have practical effects for both individual decision makers ([21], [22]) and group decisions ([23]). A number of approaches have been developed for mathematically describing human judgments. These approaches include the use of policy-capturing models in social judgment theory, probabilistic mental models, multiple-cue probability learning models, and information theory. Some researchers use a cognitive continuum theory that builds upon social judgment by distinguishing judgments on a cognitive continuum varying from highly intuitive decisions to highly analytical decisions (e.g., [24]).

Tversky and Kahneman [25] made a key contribution to the field when they showed that many of the previously-mentioned discrepancies between human estimates of probability and Bayes' rule could be explained by the use of three heuristics:

Representativeness. In the representativeness heuristic, the probability that, for example Bob is a criminal insider is assessed by the degree to which he is representative of, or similar to, the stereotype of criminal insiders. This approach for estimating probability can lead to serious errors because similarity, or representativeness, is not influenced by several factors that should affect determination of probability.

Availability. There are situations in which an information security executive conceptualizes the frequency of a class or the probability of an event by the ease with which past instances or occurrences can be brought to mind. For example, an information security executive may assess the risk of disclosure of information among financial institutions by hearing about such occurrences from one's acquaintances. Availability is a useful clue for assessing frequency or probability, because instances of large classes are usually recalled better and faster than instances of less frequent classes. However, availability is affected by factors other than frequency or probability, e.g., systematic non-reporting or underreporting of system penetrations within an industry. Consequently, the reliance on availability can lead to biases.

Adjustment and anchoring. In many situations, information security executives make estimates by starting from an initial value that is adjusted to yield the final answer. The initial value, or starting point, may be suggested by the formulation of the problem, or it may be the result of a partial computation. In either case, adjustments are typically insufficient. That is, different starting points yield different estimates, which are biased toward the initial values.

The notion of heuristics and biases has had a particularly formative influence on decision theory. A substantial body of work with applications in medical judgment and decision making, affirmative action, education, personality assessment, legal decision making, mediation, and policy making has emerged that focuses on applying research on heuristics and biases ([26]).

7 Prospect Theory

Among the different decision theories that we investigated, Prospect Theory by Amos Tversky and Daniel Kahneman (who won the 2002 Nobel Prize in Economics for its development) – best describes the behavior of insiders.

Prospect Theory distinguishes two phases in choice processes: framing and valuation ([27]). In the framing phase, the insider constructs a representation of acts, contingencies, and outcomes that are relevant to the decision. In the valuation phase, the insider assesses the value of each prospect and chooses accordingly.

From the cases that we discussed with our interviewees we found that decision theories based on the expected utility theory–where risk aversion and risk seeking are determined solely by the utility function–do not adequately explain the risk taking behavior of insiders. Insiders normally make decisions based on change of wealth rather than total gain–a behavior that is well explained by Prospect Theory. This also correlates with our model, in that insiders may not fully understand the risks of a crime that might be immensely favorable if successful.

This finding is also consistent with the results of some previous studies. For example Wood [28] finds insiders to be risk averse and their ultimate fear is to be discovered before they have mounted a successful attack. Risk aversion is the reluctance of an insider to accept a bargain with an uncertain payoff rather than another bargain with more certain, but possibly lower expected payoff. Expected value maximization is problematic in framing an insider's decision because it does not allow decision makers to exhibit risk aversion.

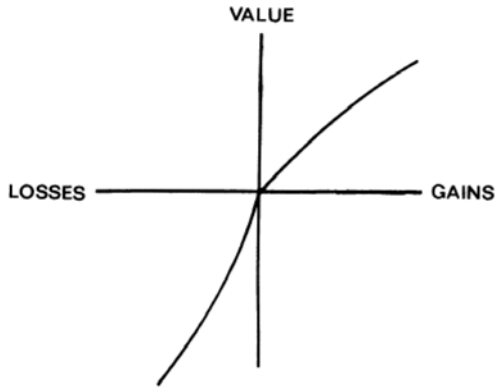
Prospect Theory has been successful in explaining individual differences that have been observed in the laboratory and outside the laboratory studies ([29]; [30]; [31]). However, some studies do not completely support applications of Prospect Theory in the real world ([32]; [33]).

Following Kahneman and Tversky, we can parameterize the value function in Prospect Theory as a power function (see Figure 2):

$$V(x) = \begin{cases} x^{\alpha} & x \geq 0 \\ -\lambda(-x)^{\beta} & x < 0 \end{cases}$$

Where $\alpha, \beta > 0$ measure the curvature of the value function for gains and losses, respectively, and k is the coefficient of loss aversion. Thus, the value function for gains (losses) is increasingly concave (convex) for smaller values of $\alpha(\beta) < 1$, and loss aversion is more pronounced for larger values of $\lambda > 1$. Tversky and Kahneman estimated median values of $\alpha = \beta = .88$, and $\lambda = 2.25$

Fig. 2. Value function from Prospect Theory (adopted from [27])



among their sample of college students. The degree of curvature of the value function represents the insider's sensitivity to increasing units gained or lost.

Expected utility theory and most normative models of decision making under risk assume the principle of description invariance: Preferences among prospects should not be affected by how they are described. Decision makers act as if they are assessing the impact of options on final assets ([31]). Prospect Theory acknowledges that choices are influenced by how prospects are cognitively represented in terms of losses versus gains and their associated probabilities–this characteristic of Prospect Theory explains the influence of perceptions on insider decisions.

We argue that the significant ability of Prospect Theory in framing and editing operations, compared to other decision theories, best describes the behavior of insiders.

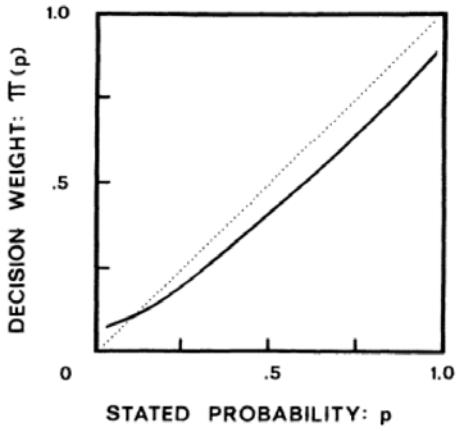
The Weighting function in Prospect Theory can be shown as follow:

$$w(p) = \frac{\delta p^{\gamma}}{\delta p^{\gamma} + (1-p)^{\gamma}}$$

Where $\delta > 0$ measures the elevation of the weighting function and $\gamma > 0$ measures its degree of curvature. Figure 3 represents shape of this weighting function:

The inverse-S-shaped weighting function is characterized by a tendency to overweight low probabilities and underweight moderate to high probabilities. Although the shape of the value function implies risk aversion for gains and risk seeking for losses, this pattern seems to be reversed for low-probability events and reinforced for high-probability events.

Fig. 3. Weighing function from Prospect Theory (adopted from [27])



8 Summary and Conclusion

This paper describes on the role of perceptions of risk and benefit of insiders in taking actions such as theft of information, sabotage, and fraud in organizations. We use the theoretical foundation of perception of risk built by Baruch Fischhoff, Paul Slovic, and of behavioral economics by Daniel Kahneman and Amos Tversky. We identify consequences and understanding as two main characteristics of perceived risk by insiders. We contend that perceived benefit plays an important role in insider decisions and that classical decision theories cannot adequately explain insider behavior.

Making effective decisions to confront insider threats requires understanding insiders' risk taking behavior and their decision heuristic. We believe that there is significant value to including risk perception management as part of a comprehensive security plan. Technical controls continue to be important, especially when coping with outsider attacks and unexpected failures. However, not all security problems can be addressed with IT-based defenses. Our research results provide one more approach to defending important computing assets against insider misuse.

9 Acknowledgments

This material is based in part upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of

Homeland Security, the I3P, or Dartmouth College. Sponsors of the Center Education and Research in Information Assurance and Security (CERIAS) also supported portions of this work. The authors would also like to acknowledge the contribution of Mr. William Keck in literature review.

References

1. Brackney, R.C., Anderson, R.H.: Understanding the insider threat. Proceedings of a March 2004 Workshop, RAND Corporation (2004)
2. Greitzer, F.e.a.: Combating the insider cyber threat. IEEE Security and Privacy, pp. 61-64. (2008)
3. Maloof, M., Stephens, G.: Elicit: A system for detecting insiders who violate need-to-know. Lecture Notes in Computer Science, 4637, pp.146-166 (2007)
4. Stolfo, S., Bellovin, S., Hershkop, S., Keromytis, A., Sinclair, S., Smith, S.: Insider attack and cyber security. Advances in Information Security, Springer (2008)
5. Bishop, M., Gates, C.: Defining the insider threat. Proceedings of the Cyber Security and Information Intelligence Research Workshop, article 15 (2008)
6. Stamper, C.L., Masteson, S.: Insider or outsider? how employee perception of insider status affect their work behavior. Journal of Organizational Behavior, 23, pp. 875-894 (2002)
7. Fischhoff, B., Slovic, P., Lichtenstein, S., Read, S., Combs, B.: How safe is safe enough? a psychome-tric study of attitudes towards technological risks and benefits? Policy Sciences, 9(2), pp. 127-152 (1978)
8. Slovic, P.: Perceptions of risk. Science, 236, pp.280-285 (1987)
9. Stoneburner, G., Gougen, A., Feringa, A.: Risk Management Guide for Information Technology Systems. NIST SP800-30 (2002)
10. Bloom, B.S., Krathwohl, D.R.: Taxonomy of educational objectives: The classification of educational goals, by a committee of college and university examiners. Handbook 1: Cognitive domain, New York, Longmans (1956)
11. Farahmand, F., Atallah, M., , Kensynski, B.: Incentives and perceptions of information security risks. Proc. of the Twenty Ninth International Conference on Information Systems, Paris (2008)
12. Wells, J.T.: Principles of Fraud Examination. John Wiley & Sons (2005)
13. Finucane, M.L., Alhakami, A., Slovic, P., Johnson, S.M.: The affect heuristic in judgments of risks and benefits. Journal of Behavioral Decision Making, Vol. 13, pp. 1-17 (2000)
14. Zajonc, R.B.: Feeling and thinking: Preferences need no inferences. American Psychologist, Vol. 35, pp.151-175 (1980)
15. Kahneman, D., Lovallo, D.: Timid choices and bold forecasts: A cognitive perspective on risk taking. Management Science, Vol. 39, No. 1, pp. 17-31 (1993)
16. MacGregor, D.G.e.a.: Perception of financial risk: A survey study of advisors and planners. Journal of Financial Planning, Vol. 12 Issue 8, pp. 68-86 (1999)
17. Lichtenstein, S., Slovic, P.: Reversals of preference between bids and choices in gamble decisions. Journal of Experimental Psychology, Vol. 89, No. 1, pp. 46-55 (1971)
18. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press (1947)
19. Savage, L.J.: The Foundations of Statistics. John Wiley & Sons (1954)

20. Kenney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley & Sons (1976)
21. Kahneman, D., Slovic, P., Tversky, A.: Judgment under uncertainty; heuristics and biases (1982)
22. Heath, L., Tindale, R., Edwards, J., Posavac, E., Bryant, F., Henderson-King, E., Suarez-Balcazar, Y., Myers, J.: Applications of Heuristics and Biases to Social Issues. Plenum Press (1994)
23. Paese, P.W., Bieser, M., Tubbs, M.E.: Framing effects and choose shifts in group decision making. Organizational Behavior and Human Decision Processes, 56, pp. 149-165 (1993)
24. Hammond, K.R.: Naturalistic decision making from a brunswikian viewpoint: Its past, present, future. In G. A. Klein, J., Orasanu, R., Calanrewood, Zsambok, E., (Eds.) Decision making in action: Models and Methods (pp. 205-227). Norwood, Albex (1993)
25. Tversky, A., Kahneman, D.: Judgment under uncertainty: Heuristics and biases. Science, 185, pp. 1124-1131 (1974)
26. Lehto, M.R., Buck, J.R.: Introduction to Human factors and Ergonomics for Engineers. CRC Press (2008)
27. Tversky, A., Kahneman, D.: Prospect theory: An analysis of decisions under risk. Econometrica, Vol. 47, No 2, pp. 263-291 (1979)
28. Wood, B.: An insider threat model for adversary simulation. SRI International, Research on Mitigating the Insider Threat to Information Systems - #2 Proceedings of a Workshop Held by RAND (2000)
29. Camerer, C.F.: Prospect theory in the wild. Cambridge Univ. Press, Cambridge, UK, (2000)
30. Odean, T.: Are investors reluctant to realize their losses? Journal of Finance, 53, pp. 1775-1798 (1998)
31. Trepel, C., Fox, C.R., Poldrack, R.A.: Prospect theory on the brain? toward a cognitive neuroscience of decision under risk. Cognitive Brain Research, Vol. 23, No 1, pp. 34-50 (2005)
32. Levy, M., Levy, H.: Prospect theory: Much ado about nothing. Management Science, Vol. 48, No. 10, October 2002, pp. 1334-1349 (2002)
33. Schroeder, N.J.: Using prospect theory to investigate decision-making bias within an information security context. Dept. of the Air Force Air University, Air Force Institute of Technology (2005)

Accumulating Evidence of Insider Attacks

Howard Chivers[1], Philip Nobles[1], Siraj A. Shaikh[1], John A. Clark[2], and Hao Chen[2]

[1] Department of Information Systems, Cranfield University, Shrivenham, UK
h.chivers/p.nobles/s.shaikh@cranfield.ac.uk
[2] Department of Computer Science, University of York, York, UK
jac/chenhao@cs.york.ac.uk *

Abstract. Insider attacks are often subtle and slow, posing the problem of integrating a large volume of event data from multiple sources over a long period. This paper proposes a scalable solution to combining evidence from multiple sources, by maintaining long-term estimates that nodes are subverted for each node in the system, rather than retaining event data for post-facto analysis. These estimates are then used as triggers for more detailed investigation. We identify essential attributes of event data, allowing the use of a wide range of sensors, and show how to apply Bayesian statistics to maintain incremental node estimates without global updating or normalization. The paper provides a theoretical account of the process, a worked example, and a discussion of its practical implications.

1 Introduction

Insider attacks pose a particular threat because of the knowledge, access, and authority of their perpetrators [12]. Such attacks often involve violations of physical or operational security, or the misuse of authority; they may also involve electronic attacks, in which case the 'electronic insider' is as big a threat as a person. It may be safer for a sophisticated external attacker to subvert an electronic system, often via social engineering, than directly subvert an employee. Such attackers may use technical means to camouflage an attack, such as indirection or address spoofing [1]; however, their most potent weapon in avoiding detection is patience – the world's largest credit card fraud was achieved with a subverted internal system that avoided discovery for over 17 months [9].

Subtle attackers are unlikely to launch large-scale scans, or use known exploits; they will seek to avoid any action that can be immediately identified as an attack. However, they are likely to cause minor security events: an attacker may test known passwords, probe for services, or test new exploits, expecting to hide within the background of user errors, mistakes and other 'noise'. The problem of detecting such an attacker is therefore one of accumulating relatively

* D. Chadwick, I. You and H. Chang (Eds.): Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST2009), Purdue University, West Lafayette, USA, June 16, 2009.

weak evidence over a long period. This issue is one of the 'grand challenges' of the internal attacker problem: "to combine events from one or more sensors, possibly of various types" while "reduce[ing] data without adversely impacting detection" [3]. This paper provides a solution to this critical problem.

The work presented here is couched in terms of networks and systems, and the identification of a subverted node, which is part of a system that is used by a corrupt insider, or is acting as an electronic insider for some other party. However, the approach to characterizing and combining diverse sources of weak evidence is equally applicable to other problems in the insider space, such as identifying criminal or espionage threats from behavioral indicators.

This paper provides a process for combining evidence from various sources based on the application of Bayesian statistics, identifies attributes that must be available to allow the combination of evidence from different types of sensor, and demonstrates the approach with a simulated slow-attack on a network.

The paper is organized as follows: Section 2 is overview of the proposed approach, section 3 describes related work, and the evidential accumulation process is developed in section 4. Section 5 presents an example to show that this process is well behaved in simple cases, while section 6 simulates a challenging insider detection problem, and contrasts the evidence accumulation process with a common, but naive, alternative approach. Section 7 discusses results and open issues, and the paper is concluded in section 8.

2 Overview

Consider how a human investigator might approach the problem of accumulating evidence in the network of Figure 1. The network consists of nodes *(A...J)* with interconnectivity as shown. Two minor security events are detected *E1,* and *E2*; they may originate from an operating system alert, intrusion detection system, or other form of event detection (see section 3).

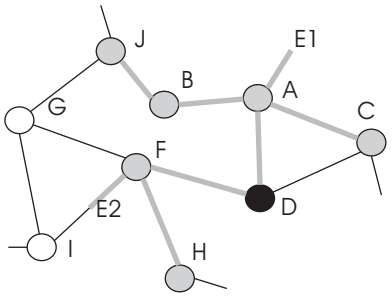


Fig. 1. Intersecting Evidence

Given information about event *E1* and the traffic in the network at the time, the investigator may determine that the nodes most likely to have originated the

event are *J, B, A, C or D*. Similarly, when *E2* occurs, at a much later date, the possible originating nodes are *D, F* and *H*. Intersecting these observations suggests node *D* as a common factor, and this may be sufficient to trigger intensive monitoring to determine if it is behaving maliciously.

The data used to identify these security events and their possible sources is necessarily transient; it may not be possible to record sufficient traffic to allow this analysis retrospectively. However, it is initially sufficient to just identify nodes that score differently; in the long, slow, game, it is only necessary to 'tip off' a further investigation by identifying one or more nodes whose behaviour may be unusual. It is not essential to record the events, the traffic from which they were identified, or even the graphs that identify possible sources, provided it is possible to somehow accumulate a 'score' for each node in the system.

This approach solves one of the critical issues in identifying slow attacks: how to maintain long-term state. Systems that try to model the behaviour of individuals, systems or protocols, are forced to retain large amounts of data, which limits their scalability. In the approach described here, the state size is a small multiple of the number of nodes in the network; this state is readily distributed, and its storage is feasible, even for organizations with global networks.

The 'score' that we propose for each node is the probability that the node is subverted, based on the application of Bayesian statistics. This naturally allows incremental updating, and translation of the problem frame from events, which are related to behaviour, to individual attackers. Simpler schemes, such as the event counting used to introduce this section, can be shown to be inferior, as demonstrated by the network simulation in section 6.

In summary, we propose that to identify subtle or inside attackers:

- The primary objective is to identify nodes for further investigation.
- Long-term state is restricted to an incremental estimate of the probability that each node is an attacker.
- Node estimates are updated following every security event, taking account of transient network information that may be available at the time of the event.

This process is complementary to conventional intrusion detection using signatures or heuristics. There is no need to gradually accumulate evidence if the attack is evident; for example, high levels of network activity due to a worm or virus provide compelling evidence of an attack, and in these cases the secondary investigation is concerned with incident management, rather than confirmation.

Section 4 describes how node scores can be estimated and maintained, following a brief summary of related work.

3 Related Work

The use of a tiered approach to insider threat detection, detection followed by a more detailed forensic investigation, is proposed by Bradford et al [4]. Users are profiled according to their function, and deviation from normal behaviour

triggers more intensive data collection. Sequential hypothesis testing is proposed to determine whether a process is anomalous and more intensive data collection should be initiated. However, the authors do not show an implementation of their approach, and remark that it could not be carried out for "every user regardless", but itself requires a "triggering process".

The problem is the volume of data that must be maintained, and this is also a issue with datamining approaches, which are often proposed as an adjunct to intrusion detection or audit. Research proposals to alleviate the scalability issue include improving the quality of the raw data, by discovering better behavioral indicators [11] or classifying input features [6], the latter using a Bayesian classifier. An alternative approach by Staniford et al [14] is to selectively retain anomalous network data, with the aim of identifying slow network scans. Anomalous packets are identified based on heuristics developed from real scans. Other approaches include statistical filtering, primarily to reduce false alarm rates and support visualization [7]. In essence, however, all these approaches require the storage of large volumes of event data for later analysis, and the authors themselves often identify scalability as a problem [11].

Aggregation as a means of detecting slow or stealthy attacks has been proposed by Heberlein [10]. His assumption is that slow attacks are still systematic, and the attacker will eventually repeat the attack many times, possibly against different targets. Alerts are classified, accumulated, and displayed on a visualization grid, and any persistent activity which raises alerts of the same type over a long period, can be identified. Although similarly motivated, our work differs by accumulating evidence of attackers, not of incidents, removing the restriction that attackers need to repeat similar attacks. Heberlein's algorithm is also a counting process, which we show to be inferior to statistical reasoning.

Other work directed toward the insider problem is focussed on characterising an attacker's behaviour. The security indicators ('events') used may range from an individual's buying and travel preferences, to electronic alerts. For example, Burford et al [5] propose a comprehensive framework of 'observables' that are used to build a model of individuals' behaviour via graph theory. Eberle et al [8] develop graphs of behavioral events, such as phone calls, to identify subgraphs of normal behaviour, which are used to search for similar but anomalous occurrences. These approaches offer the advantage of modeling the potential attacker, and providing interesting insights into observable behaviour; however, their application may be limited by the computational cost of graph matching over large datasets, as well as by data scalability.

Most of the work described above is still formative; network intrusion detection, however, is established in the literature and supported by both open and propriety products [2]. An intrusion detection system (IDS) uses a behavioral model of a system or protocol and detects anomalous events by either recognizing predefined signatures, or by heuristics. Both approaches have strengths and weaknesses, but despite the usefulness of IDSs in practice, they are hampered by a lack of scalability, and tend to generate large numbers of false positive alerts [2]. From the perspective of this paper, IDSs are an effective way of generat-

ing events which may indicate an attack, but are unable to maintain sufficient state to identify slow attacks. An IDS is not the only possible source of security events; for example, the behavioral events referenced above, operating system audit trails, and even Honeypots [13], which are security traps with no operational functionality, are all possible sources of security events.

In summary, the challenge of integrating information from many sources in order to identify patient internal attackers is still an important open question [3].

3.1 Updating Evidence

This section develops the detailed theory necessary to achieve the method outlined in section 3: to collapse the problem of attacker identification to updating a single score for each network node, or user. The section first outlines the evidential scenario, and the attributes required to characterize security events. Standard Bayesian updating is summarized, followed by the development of the process for updating evidence of insider attacks. Finally, the practical issue of relating this process to real security events is discussed.

Definitions

Node: This paper uses network terminology, without loss of generality to broader types of human or attack behavior. A node is a network component, such as a user's end system, a router, or a user.

Event: An event is an alert that indicates a possible security violation; it may be an anomalous phone call, a failed connection, or something more certain, such as a known electronic exploit.

The evidential scenario is presented in Figure 2. Node (a) is a network node, and (x) is an event which is detected somewhere in the network; there is some evidence that identifies the nodes that may have originated the event.

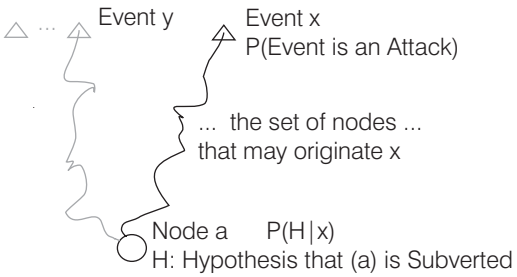


Fig. 2. Evidential Scenario

Event (x) may indicate an attack. Some security events are almost certainly attacks; however, there are many more that may be user mistakes, backscatter,

or other forms of network 'noise'. For example, an attempt to connect to a non-existent webserver is often a simple mistake, but could also be an attack probe.

In addition to uncertainty about the extent that an event is an attack, there may also be uncertainty about the source of the event. For example, the attacker may be able to spoof its network address, or the event may only be traceable to a subnetwork. In order to accumulate evidence from a wide range of different sources, they must be characterized by uniform parameters that describe these various attributes. We propose that the difference between different security events can be characterized by three parameters:

- *P(Attack):* the probability that an particular event (x) is actually caused by an intentional attack.
- *The Causal Node Set:* the set of network nodes that could have caused the event, even if it was a not an attack.
- *P(Causal):* the probability that the causal node is actually within the node set.

Given a sequence of events characterized by these parameters, we wish to investigate the hypothesis that a particular node is subverted, or acting as the agent of an attacker. We will first summarize the standard approach to Bayesian updating, then show how it can be applied in this case.

4 Bayesian Updating

Bayesian updating provides an estimate of the probability that hypothesis H is true, given an event, (x).

$$P(H|x) = \frac{P(x|H) \cdot P(H)}{P(x)} \tag{1}$$

This theorem uses *P(x|H)*, the probability of event *(x)* given that the hypothesis is true, to update the initial ('prior') estimate of the probability that the hypothesis is true, *P(H)*. Simple updating of this type is often used in medical diagnosis; given knowledge of the probability of a symptom (the Event) given a disease (the Hypothesis), it provides a principled estimate of the likelihood of the disease given the symptom. It is essentially this change of reference frame – from symptom to cause – that is needed to identify internal attackers from their behaviour.

The denominator, *P(x)*, the probability of the event, is effectively a normalising factor. In many cases, including ours, it is difficult to estimate; however, by making use of $P(H|x) + P(\neg H|x) = 1$, it is straightforward to eliminate *P(x)* and derive the following standard variant of Bayes' theorem:

$$P(H|x) = \frac{P(x|H) \cdot P(H)}{P(x|H) \cdot P(H) + P(x|\neg H) \cdot P(\neg H)} \tag{2}$$

Practical applications of Bayes theorem often combine several sources of evidence. Achieving a usable update formula for multiple evidential events requires

an assumption of conditional independence – that individual security events do not cause each other. The derivation is given in standard texts on Bayes. For example, the formulae, which gives the revised probability of the Hypothesis, *H*, given two items of evidence, *(x)* and *(y) is*:

$$P(H|x,y) = \frac{P(x|H) \cdot P(y|H) \cdot P(H)}{P(x|H) \cdot P(y|H) \cdot P(H) + P(x|\neg H) \cdot P(y|\neg H) \cdot P(\neg H)} \quad (3)$$

This pattern can be extended to cope with multiple items of evidence.

4.1 Combining Evidence from Security Events

The evidential scenario is described at the start of this section; in detail, we define:

S	The set of all nodes in the system.
#S	The total number of nodes in the system.
a,b...	Particular network nodes. $a, b, ... \in S$
$\mathbf{H}_a$	The Hypothesis that we wish to update: that a particular node (a) is subverted, or being used to mount an attack within the system.
E	The set of all Security Events generated over time.
x,y...	Particular events that may provide evidence of an attack. $x, y, ... \in E$
$\mathbf{P}_x$**(Attack)**	The probability that a particular event (x) actually originates from an intentioned attack.
$\mathbf{C}_x$	The Causal set of nodes associated with event (x); in other words, the set of nodes that may have originated the event.
$\#\mathbf{C}_x$	The number of nodes in set C_x
$\mathbf{P(C}_x\mathbf{)}$	The probability that C_x includes the node that originated the event. In other words the accuracy with which C_x is estimated.

The parameters P_x*(Attack),* C_x*,* and *P(*C_x*)* were introduced in the introduction to this section as the attributes needed to characterize an event.

We wish to update an estimate of the probability of H_a, following event (x). In equation 2, above, the prior probabilities $P(H), P(\neg H)$ will depend upon the node (e.g. the prior probability of subversion of a server may be significantly different to that of a user client), but will otherwise be constant, so we can write $P(\neg H_a) = 1 - P(H_a)$

However, to obtain an estimate of $P(x|\neg H_a)$it is necessary to take into account that it may not be possible to attribute an event to a single node (a), but only identify a set of nodes, C_x, from which the event may have originated. Unlike the prior probability, this is dependent on the event, as well as on the node, since different events will be generated by different sensors with different capabilities and views of the network.

There are three types of node to consider: the node currently being updated, (a), other nodes in the set C_x, and other nodes in the system that are not in C_x. As a consequence, for an event (x), there are two alternative hypotheses to H_a:

$\mathbf{R}_{a,x}$ That node (a) is not an attacker, but is within C_x.
$\mathbf{I}_{a,x}$ That node (a) is not an attacker, and is outside C_x.

Since $R_{a,x}$ and $I_{a,x}$ are disjoint, we can write:

$$P(x|\neg H_a) = P(x|R_{a,x}) \cdot P(R_{a,x}|\neg H_a) + P(x|I_{a,x}) \cdot P(I_{a,x}|\neg H_a) \tag{4}$$

If a node is not an attacker, we can expect the probabilities of the two alternative hypotheses to be a simple function of the numbers in each set. Substituting $P(R_{a,x}|\neg H_a) = \frac{\#C_x}{\#S}$ and $P(I_{a,x}|\neg H_a) = \frac{\#S-\#C_x}{\#S}$into equation (4), we obtain:

$$P(x|\neg H_a) = P(x|R_{a,x}) \cdot \frac{\#C_x}{\#S} + P(x|I_{a,x}) \cdot \frac{\#S - \#C_x}{\#S} \tag{5}$$

Substituting (5), and the expression for $P(\neg H_a)$given above into the normalized version of Bayes theorem given in equation (2), we obtain:

$$P(H_a|x) = \frac{P(x|H_a) \cdot P(H_a)}{P(x|H_a) \cdot P(H_a) + [P(x|R_{a,x}) \cdot \frac{\#C_x}{\#S} + P(x|I_{a,x}) \cdot \frac{\#S-\#C_x}{\#S}] \cdot [1 - P(H_a)]} \tag{6}$$

Defining:

$$\Delta_{a,x} = \frac{P(x|H_a)}{P(x|R_{a,x}) \cdot \frac{\#C_x}{\#S} + P(x|I_{a,x}) \cdot \frac{\#S-\#C_x}{\#S}} \tag{7}$$

Rearranging equation (6) and substituting in $\Delta_{a,x}$ gives:

$$P(H_a|x) = \frac{\Delta_{a,x} \cdot P(H_a)}{\Delta_{a,x} \cdot P(H_a) + [1 - P(H_a)]} \tag{8}$$

This update formulae can be extended to multiple events under the assumption of conditional independence, similar to equation (3); the resulting update formulae becomes:

$$P(H_a|x, ...y) = \frac{\Delta_{a,x} \cdot ... \cdot \Delta_{a,y} \cdot P(H_a)}{\Delta_{a,x} \cdot ... \cdot \Delta_{a,y} \cdot P(H_a) + [1 - P(H_a)]} \tag{9}$$

This is the final update formulae. Bayes updating is often presented in this form; the application specific elements are contained in the definition of $\Delta a,x$, which will now be further developed by substituting in the information that characterizes security events. We will consider each element of equation 7 in turn.

P(x|H_a) is the probability that an event (x) occurs given that the identified node is actually subverted. Since we will update in response to events, then for each event $P_x(Attack)$, provides a plausible value, which only discounts the possibility that false alarms may have originated from a genuine attacker.

$P(x|R_{a,x})$ is the probability that the node is not subverted, but that it lies within the set of nodes that may have originated the attack. In terms of our model parameters, this corresponds to $[1 - P_x(Attack)] \cdot P(C_x) \cdot \frac{\#C_x}{\#S}$

$P(x|I_{a,x})$ is the probability that the node is not subverted, but that it lies outside the set of nodes that may have originated the attack. Of course, we would wish to correctly identify the range of nodes that may originate the attack, in which case this probability will be zero; however, in the real world we must take into account the possibility that C_x cannot be enumerated with certainty, giving a value of: $[1 - P_x(Attack)] \cdot [1 - P(C_x)] \cdot \frac{\#S - \#C_x}{\#S}$

Substituting these parameters into $\Delta_{a,x}$, we obtain:

$$\Delta_{a,x} = \frac{P_x(Attack)}{[1 - P_x(Attack)] \cdot \left[P(C_x) \cdot \left(\frac{\#C_x}{\#S} \right)^2 + [1 - P(C_x)] \cdot \left(\frac{\#S - \#C_x}{\#S} \right)^2 \right]} \tag{10}$$

This ratio is used to update the probability that a node is an attacker.

4.2 Updating in Practice

Equations (9) and (10) provide the necessary theory to achieve the objective of discarding the details of security events, while retaining a simple score for each node which summarises the evidence that the node is an attacker. A naive algorithm to achieve this would be:

1. Initialize each node with its prior probability, $P(H_a)$. Values may be suggested by survey data; in large systems a prior assumption that a small number of nodes (e.g. 10 of 10,000) are likely to be subverted is reasonable. This parameter is of most value if different nodes have significantly different prior probabilities, for example the difference between a router and a laptop.
2. For each security event:
   (a) Establish the distinguishing parameters (the probability that it is an attack, the nodes that may have originated the attack, and the probability that the node set contains the attacker).
   (b) Calculate Δ from equation (10).
   (c) Multiply the node score by Δ for each node in the set, but not for any others in the system.
3. When required, substitute the node score (the product of all Δs) into equation (9) to obtain the probability that the node is an attacker.

A feature of accumulating evidence in this way is that, assuming the evidence collected is generally useful (a true positive is more likely than a false positive), then over a long period the probabilities converge towards unity. However, we are only concerned with comparative scores, in order to identify nodes that are distinctive and require further investigation. In practice, then, it is sufficient to use Logarithmic scores, simply adding $\text{Log}(\Delta)$ to each node indicated by an

event. Equation (9) can still be reconstructed from this information, but more usually, the highest node score is chosen for further investigation.

The reader may be wondering about the value of calculating Δ at all at this stage, since we simply add its logarithm to the score for indicated nodes. However, this differs significantly from a simple counting algorithm, where the score for each node is incremented when it is identified as the possible source of a security event. The update value, Δ, characterizes exactly how much evidence is provided by each event. This important distinction is illustrated in the worked example presented in section 6.

5 Simple Example

Before showing a simulation of a realistically difficult example (see section 6), this section explores if the evidential accumulation process has intuitively appealing behaviour; in particular, given a single sub-network, in which the sender can be readily identified:

- Does the evidential process identify an attacker sending at a slightly higher rate than the background of errors from normal nodes?
- If the rate of attack increases, is the process stable, and does it enable the attackers to be identified earlier?
- Does the process accommodate multiple attackers with different rates of attack (i.e. can one node hide behind another's attack)?

We assume a single sub-net of 50 nodes, in which the originating node of an event can be identified (i.e. C_x=1); we assign $P(Attack)$ an arbitrary probability of 0.083. Time is divided into slots (e.g. single minutes) and the average background rate of random innocent events that may be misinterpreted as attacks is 1/50 per node – in other words, one event per minute. Three nodes within the sub-net are designated attackers, and they generate random attacks at rates of 2, 4 and 8 times the total background rate.

The scores resulting from simulating this scenario are shown in Fig. 3. All three attack nodes are well distinguished from the background level of events, which is indicated by the 'other nodes' result, which is the score for a typical innocent node. As would be expected, if the attack rate is higher, the discrimination improves. The accumulation of evidence is well behaved, and the higher rate nodes do not interfere with the accumulation of evidence relating to attackers operating at a lower rate.

6 Insider Attack Simulation

This section presents an example of evidence updating in practice. The example shows that complex network propositions can be accommodated straightforwardly, and contrasts the principled accumulation of evidence with a simple counting scheme. This example includes features that are common in this problem space, including:

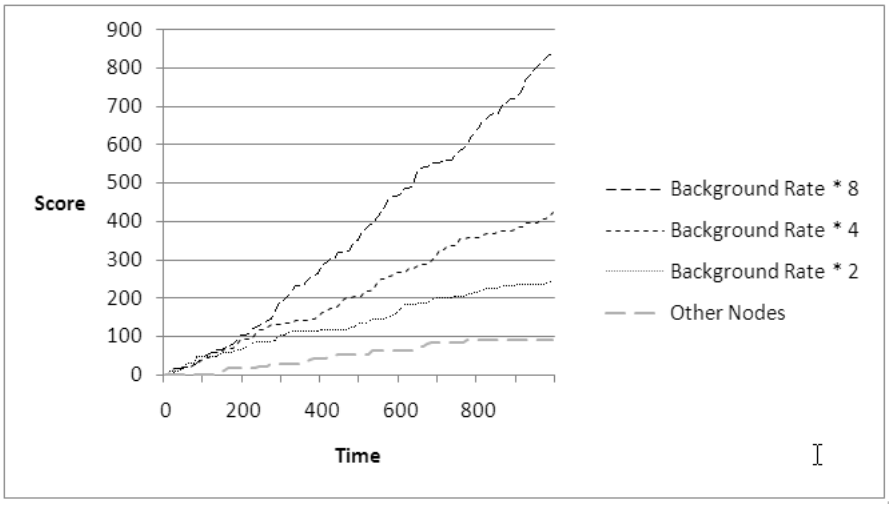


Fig. 3. Simple attacker scenario in a single sub-network

- Sensors with different capabilities; for example, certainty of detection and ability to identify source nodes.
- Attackers whose rate of attack is below the background rate of false positive alerts for the system.
- Attacks that employ address spoofing.

An important practical issue is the estimation of the three parameters that characterize a security event; relating these to actual systems and assessing the need for accuracy is subject to ongoing study. To date it has been possible to achieve realistic results by assigning *P(Attack)* as a fixed value for a given sensor within a deployment context, and by creating a simple rule-set that maps the network connection associated with an event to a set of nodes, giving C_x and $P(C_x)$, depending on the configuration and protocol.

The network used in this example is given in Fig. 4. This network has 200 nodes, most of which are user systems located in four separate client sub-networks. Two of these sub-networks have nodes that been subverted and are attacking the system. The purpose of dividing the clients into several sub-nets (apart from the fact that this is a common configuration) is to contrast the detectability of attackers in different sized sub-networks, given that we assume that in many cases it will be possible to identify only the sub-net from which an attack originated. This arrangement allows us to investigate the scores accrued for an attack node (3 or 53) versus other nodes in the same sub-net, and nodes in a control sub-net of the same size and performance with no attacker.

Most of the traffic in the system is between the clients and servers, via the core network. Router and firewall detail is not shown, and because the object is to investigate evidence accumulation rather than event generation we model a two unspecified types of security event: those that can be detected within client

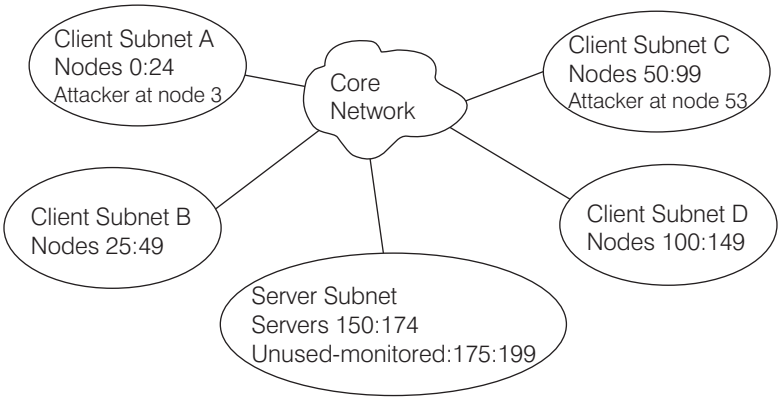


Fig. 4. Test Network

sub-networks, and events in the server farm. For example, an event could be an attempt to connect to an exploitable network port.

Attackers are expected to generate security events at a rate that is much lower than the background rate of 'mistakes' by normal clients, in order to remain undetected. In the simulation below, time is measured in arbitrary clocks (e.g. minutes), and the probability of a normal client generating a security alert in any time slot is 1/150; in other words the system suffers an average of one false alarm every minute. In contrast, attackers generate events at a rate of 1/25; one event every 25 minutes.

In addition to the low attack rate, to further avoid detection, attackers use address spoofing. Events detected outside the sub-net containing the attacker can only be assigned to the whole sub-net. Only events identified within the sub-net containing the attacker (i.e. directed toward nodes within that sub-net) can be traced to a specific node.

An outline calculation illustrates the difficulty of this problem. Consider the attacker in sub-net A. Viewed from outside, the sub-net can be expected to generate innocent background events (false alarms) at a rate of one every 6 minutes (*P()=25 * 1/150*). The events generated by the attacker are distributed at random across the network, so of these, 25/200 are towards the attacker's own sub-network, and 175/200 are visible externally. This results in an externally visible attack every 29 minutes (*P()=1/25 * 175/200*), and these events can only be identified with the whole sub-net. Events targeted at random to nodes within the sub-net can be identified to a particular attacker, but these occur at a rate of only one every 200 minutes (*P()=1/25 * 25/200*). Of course, given this information the reader could devise a solution to identify the attacker, but the problem addressed here is how to use all the available information when the location of the attacker and the traffic patterns are unknown in advance.

In summary, the event parameters used in the simulation are:

C_x contains all the nodes in the source sub-net, unless the destination of the network message that caused the event is in the same subnet as the source, in which case C_x contains just the source address.

$P(C_x)$ is set to unity, since C_x includes all the possible source nodes.

$P(Attack)$ is set to 0.33 for all locations except the server nodes, for which a value of 0.083 is assigned. (These are arbitrary, for the sake of demonstration. It seems plausible that an incident at a location to which most of the traffic is directed is less likely to be an attack, but in practice that is dependent on the actual event. The only special feature in the choice of value is avoiding fractions such as 25/150 that match the system topology and may produce anomalous results in a small system. Varying these parameters result in different scores, but not at the expense of overall discrimination.)

A network simulator was used to generate random traffic as outlined above, and the scores for the resulting security events were accumulated as described in section 4. The results are shown in Fig. 5.

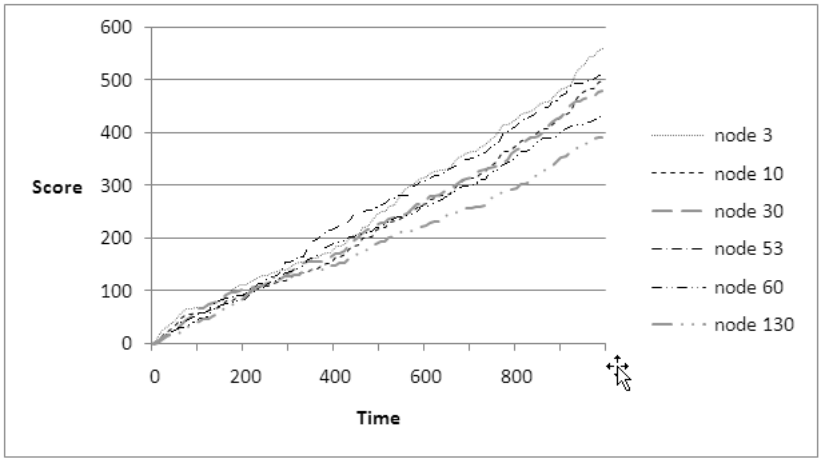


Fig. 5. Network Simulation Results

Fig. 5. shows node scores as they are accumulated. The nodes shown are attackers (3,53), representative nodes in the same sub-nets (10,60), and representative nodes in the same sized sub-nets with no attackers (30,130). Nodes (3,10,30) are from 25-node sub-nets, and nodes (53,60,130) are from 50-node sub-nets, which contain a significant proportion of the nodes in the network.

The results show that insider attacks can be clearly distinguished from background noise in the system. A longer running simulation, given in Fig. 6., provides a view of the asymptotic performance of the process.

For each size of sub-net the proposed scoring clearly distinguishes the attacker as an individual, and the sub-net containing the attacker, from the control sub-

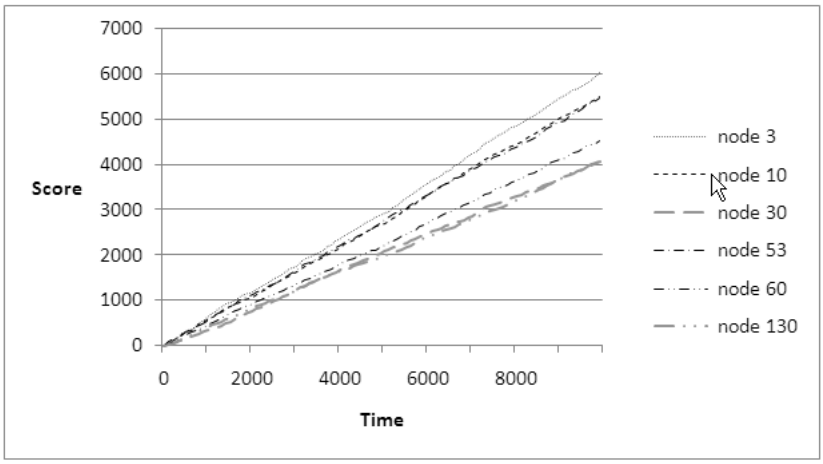


Fig. 6. Long Term Performance

net with no attacker. The distinction between different sized sub-nets is not fully maintained: both attackers are well distinguished from both control networks, however, the smaller sub-net containing an attacker is not well distinguished from the individual attacker in the larger sub-net. In practice, grouping the small sub-net with the two attackers does not present a problem, since it still provides a correct diagnosis of the attacks, that can be subject to further investigation. We conjecture that the issue here is not that the scoring method is inherently biased between different sizes of C_x (that is, any more than the actual evidential content varies with C_x), but that the larger sub-networks in this system are a substantial fraction of the system size.

The effectiveness of this Bayesian approach can be judged by comparison to the counting algorithm used to introduce section 2, and adopted by some researchers. Assuming that the same deductions can be made from the security events (specifically that C_x is the same for each event), the result of using a counting approach, where node scores are simply incremented if they are identified, is given in Fig. 7.

On a realistic problem, the counting approach fails in almost every respect. Attackers are not distinguished from other nodes in their sub-net, and there is little difference between a sub-net containing an attacker, and a control sub-net with no attacker. Instead, the primary distinction is between nodes on the basis of network size; essentially the larger sub-nets generate more background traffic, so receive a proportionately higher score.

7 Discussion

The proposed updating process is effective because it relates event evidence to the hypothesis that the node (or user) is an attacker. This change of reference

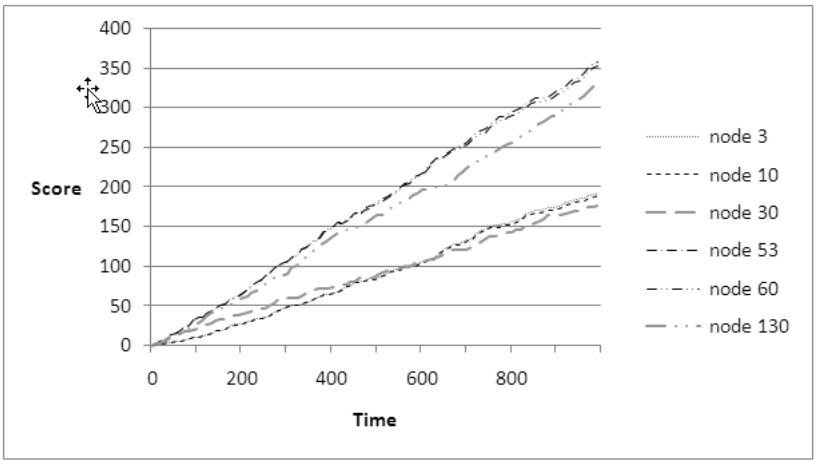


Fig. 7. Counting Algorithm Performance

frame allows event data to be discarded, while retaining the weight of evidence for attackers. The process scales linearly with the number of nodes in the system, and is likely to be applicable to a very wide range of systems and circumstances.

The updating ratio, Δ, can be thought of as the ratio of true positives to false positives. However, Bayes has been used, rather than the simple probability ratios that would be suggested if information theory was employed, in order to effect the change of viewpoint from the event to the attacker. Δ takes account of ancillary information such as the number of nodes that are indicated by the event, and the degree of certainty in their estimation.

Δ can be used as a figure of merit for sources of information; essentially, if Δ is consistently fractional for a sensor, then the resulting events will degrade the quantity of available information, rather than improve it.

The attributes described in section 4 (probability of attack, possible sources, and likelihood that the attacker is in this set) are not specific to any particular type of event generator, and can be applied at different levels of abstraction, if necessary within the same system.

There are a number of practical considerations that are subject to ongoing study. The first implementation decision is which real components are regarded as 'nodes': should nodes model all network components, just routing components and endpoints, or just endpoints such as clients, servers or users? To date, only endpoint nodes have been considered; this decision is based on the prior probability of network components originating attacks, and the convenience in associating events with their possible sources.

A key practical issue is how to determine which nodes are a potential source of any particular event, and to what degree. Ideally this assessment would be evidence-based using recent network history, but although this is feasible in principle, it is an open question if this can be achieved in practice. However,

even simple strategies, such as the one used in section 6, provide demonstrable benefit.

This research is ongoing, and other open issues include the sensitivity of the assignment of $P(Attack)$ for disparate sensors, and the possibility of decision criteria other than the maximum score function used above.

8 Conclusion

This paper provides a solution to a critical problem in insider attacker discovery: how to combine events from multiple sensors, and manage the data explosion that is otherwise needed to support the identification of long-running attacks.

The key concept is to move away from maintaining models and evidence of behaviour, and instead maintain an incremental assessment for every user/node in the system that the node is an attacker. This approach is extremely scalable; the updating algorithm is soundly based in Bayesian statistics, and avoids the need for global updating or normalization. The approach is well behaved, in the sense that higher volumes of attack make detection easier, and in a worked example which includes several of the difficulties faced in practice, it significantly outperforms counting algorithms (see section 6).

In addition, this work identifies the attributes or parameters that need to be standardized for disparate sources of security event to be combined, allowing the use of a wide range of different sensors, at different levels of abstraction. The key criteria for a sensor (see section 7) is that it tends to provide information rather than add confusion, and a side effect of the updating process presented here is a criteria for deciding when this is the case.

Research on this approach is ongoing, both using simulation and relating the work to real sensors; some of the open questions are described in section 7.

References

1. CERT incident note IN-98-05: Probes with spoofed IP addresses, 24 November 1998.
2. Rebecca Bace and Peter Mell. Intrusion detection systems (IDS). Technical Report SP 800-31, National Institute of Standards and Technology (NIST), 2001 2001.
3. Richard C. Brackney and Robert H. Anderson. Understanding the insider threat. Technical Report Proceedings of March 2004 Workshop, RAND National Security Research Division, 2004.
4. Phillip G. Bradford, Marcus Brown, Josh Perdue, and Bonnie Self. Towards proactive computer-system forensics. In *International Conference on Information Technology: Coding and Computing (ITCC 2004)*, pages 648 – 652. IEEE Computer Society, 2004.
5. John F. Buford, Lundy Lewis, and Gabriel Jakobson. Insider threat detection using situation-aware MAS. In *11th International Conference on Information Fusion*, pages 1–8, Cologne, Germany, 2008. IEEE Xplore.
6. Srilatha Chebrolua, Ajith Abrahama, and Johnson P. Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers and Security*, 24(4):295–307, 2004.

7. Jeffrey B. Colombe and Gregory Stephens. Statistical profiling and visualization for detection of malicious insider attacks on computer networks. In *The 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 138–142. ACM Press, 2004.
8. William Eberle and Lawrence Holder. Insider threat detection using graph-based approaches. In *Cybersecurity Applications & Technology Conference For Homeland Security (CATCH)*, pages 237–241. IEEE Computer Society, 2009.
9. Dan Goodin. TJX breach was twice as big as admitted, banks say. The Register, 24 October 2007.
10. Todd Heberlein. Tactical operations and strategic intelligence: Sensor purpose and placement. Technical Report TR-2002-04.02, Net Squared, Inc., 9 September 2002 2002.
11. Nam Nguyen, Peter Reiher, and Geoffrey H. Kuenning. Detecting insider threats by monitoring system call activity. In *2003 IEEE Workshop on Information Assurance*, pages 18–20, United States Military Academy, West Point, 2003. IEEE Computer Society.
12. Marisa Reddy Randazzo, Dawn Cappelli, Michelle Keeney, Andrew Moore, and Eileen Kowalski. U.S. secret service and CERT coordination center/SEI insider threat study: Illicit cyber activity in the banking and finance sector. Technical report, Software Engineering Institute, Carnegie Mellon University, August 2004.
13. Lance Spitzner. Honeypots: Catching the insider threat. In *19th Annual Computer Security Applications Conference (ACSAC '03)*, pages 170–179. IEE Computer Society, 2003.
14. Stuart Staniford, James A. Hoagland, and Joseph M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

A Exploratory Study on R&D Strategies in Industrial Technology Security

Hangbae Chang[1], Jonggu Kang[1] Hyukjun Kwon[2], and Ilsun You[3]

[1] Daejin University, San 11-1, Sundan-Dong, Gyeonggi-Do, 487-711, Korea
hbchang@daejin.ac.kr and Jaikang7@gmail.com
[2] Yonsei University, New Millenium Hall, 262 Seongsanno, Seodaemun-Gu, Seoul, 120-749, Korea
junkwon@yonsei.ac.kr
[3] Korean Bible University, 205, Sanggye-Dong, Nowon-Gu, Seoul, 139-791, Korea
isyou@bible.ac.kr
*

Abstract. To enhance international competitiveness through the protection of cutting-edge industrial technology, it is essential to establish the policy for strengthening ability to develop industrial security technology and raising international competitiveness. In this study we investigated and analyzed not only the ecumenic trend but also the present condition, then we executed the deduction of the industrial security technology development program in a aspect of government and analyzed the current status of the technical security technology for developing security technology and increasing leaks of the advanced industrial technology.

1 Present Status of Industrial Technology Leakage

According to the survey conducted by National Intelligence Service in 2008, the number of disclosure of domestic industrial technology leakage is 125 from 2000 to December of 2007. If these cases were not detected, it could have caused approximately 95 trillion won of property loss. If we have a look at the status of annual industrial technology leakage disclosure, the number of attempts to thieve technology which were less than 10, but it has recorded 26 in 2004, 29 in 2005, 31 in 2006, 32 in 2007. It indicates a constant increase and is urgent to prepare a strategy to prevent the technology leakage.

The main subject of industrial technology leakage is primarily divided into internal and external stakeholders[9]. The industrial technology leakage by insider which targets important information or electronic documents occurs via personal computer, web based e-mail, and internet messenger[1][2]. And in case of offline documents, it was reported as they are flowed out through Web, trespass by outsider committing system hacking with virus or warm, larceny by outsider flowing

* D. Chadwick, I. You and H. Chang (Eds.): Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST2009), Purdue University, West Lafayette, USA, June 16, 2009.

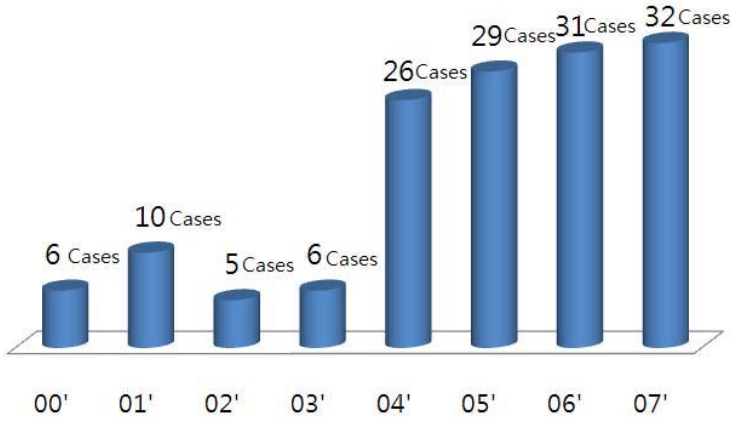


Fig. 1. Status of Industrial Information Leakage

out offline documents produced by printer or photocopier. There exists an actual case that outsider for maintenance accessed database of business process system and flowed the large amount of information and offline documents out.

Likewise, to prevent an industrial technology leakage, domestic authority concerned put Technology Leak Prevention and Industrial Technology Protection and Support Act in operation to improve the competitiveness of domestic industrial and contribute to development of national economy by preventing illegal leakage of the industrial technology. Yet for a concrete application of this Act, it is essential that the current status of industrial technology security and further study of this field is needed. Thus in this study, we analyzed the current level of domestic industrial security technology and technical competitiveness. We expect to utilize analysis data as basic information for improving international competitiveness and ability to develop industrial security technology[6].

To execute this plan, we analyzed the needs for industrial technology protection and designed the technical framework to fulfill those needs which were deduced. Following designed framework, we analyzed a current level of technology and limitation then deduced further development subject[8][10].

2 Investigation of Needs for Industrial Security Technology

In this study, to investigate actual needs for industrial security technology, we visited 15 providers of technology and 15 demander of technology then conducted in-depth interviews. The primary needs for industrial security technology are as followings[4][5]:

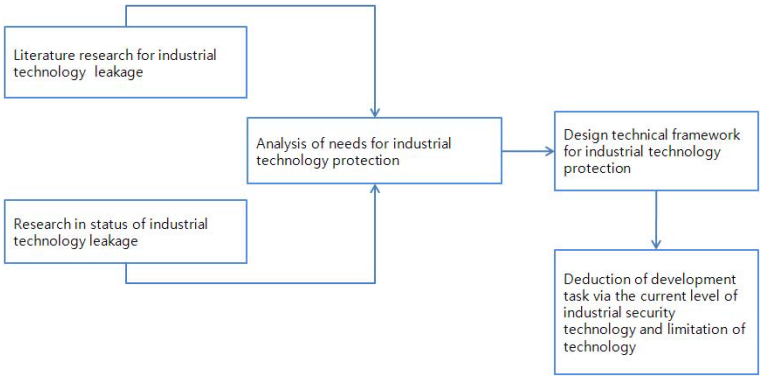


Fig. 2. Research Methodology

- As a result of the investigation, it appeared to be essential to develop counter measures for emergence of various portable storage devices(secure digital card, compact flash card, memory stick) and communication methods(infrared data communications, wireless internet, blue tooth, etc)
- Some security technologies for ordinary business documents(word, excel, power point files) have reached secure level, but security technologies for blueprints or program source documents have yet to be well developed
- The access control method is mainly used for database security technology rather than encryption due to a performance problem and there exist needs for some technology enabling illegal SQL questions to be standardized.
- Measure model for security level of remote computer is still on the way of development. And further researches about control method and resource utilization authority management for computers which reached some extent of security level.
- Currently, there occurs some security vulnerable spot in the linked section because there isnt the integration between physical and technical security.

3 Technical Industrial Security Technology Framework Design

In this study, according to disadvantage analysis result derived from risk analysis process, we applied industrial security technology design methodology based on risk analysis for solving vulnerability[3]. Information security technology development methodology based on risk analysis listed vulnerability and threats for information asset through information asset identification and analysis. Then we designed technical industrial security technology framework by reflecting assessment result about influence and risk caused by certain attack to needs for security technology development.

Before anything else, the patterns of the security vulnerability of the personal computer are classified as the damage of internal information in personal com-

puters caused by malicious external access(outflow of document file by hacking tool considering the vulnerability of operation system, virus, worm), unreliability(external penetration according to the absence of window password during booting, outflow of document file caused by the absence of screen saver) of personal computer(access control) management, and intentional internal documentation leakage by personal computer user(via e-mail, portable storage device).

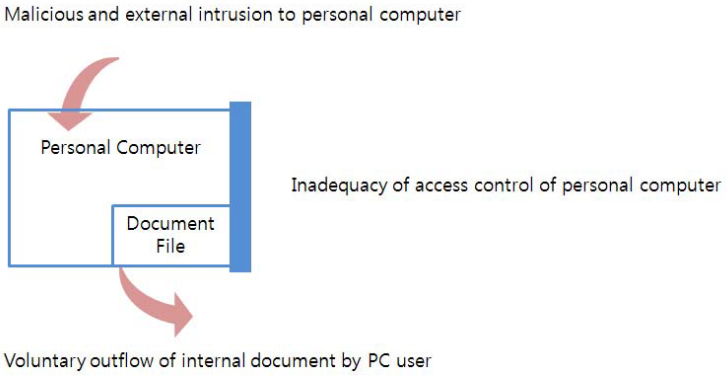


Fig. 3. Patterns of security vulnerability of PC

The patterns of the vulnerability of electronic document are classified as unencryption(circulation of the unclassified confidential document) and ungraduation, inadequacy of access control in a way of reading, editing, conveyance, and printing of the documents(abuse of users' authority, illegal outflow via e-mail and portable storage devices, theft and loss), and illegal use of destructed document(undestruction after using document, illegal outflow of document by restoring deleted document)

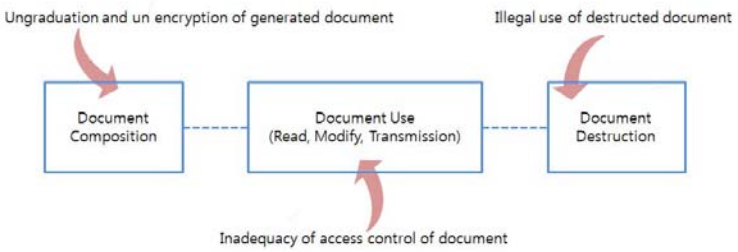


Fig. 4. Patterns of inadequacy of access control for document

The patterns of the vulnerability of database are classified as indiscreet access to database(read or outflow unrelated data file, abusing access authority) of server administrator(or usual user), outflow of data file peculating access authority), outflow of data file by peculating access authority of database(outflow of data file by peculating id and password of user or administrator), and information damage caused by the malicious penetration from outside of the organization to server or database.

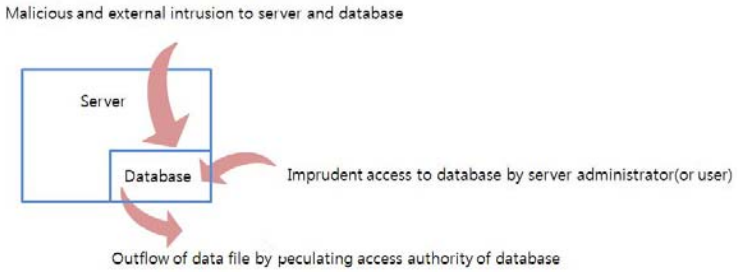


Fig. 5. Patterns of vulnerability of database

Lastly, the patterns of the vulnerability of network are classified as packet sniffing, penetration utilizing the vulnerability of network equipment, and network pulse sniffing.

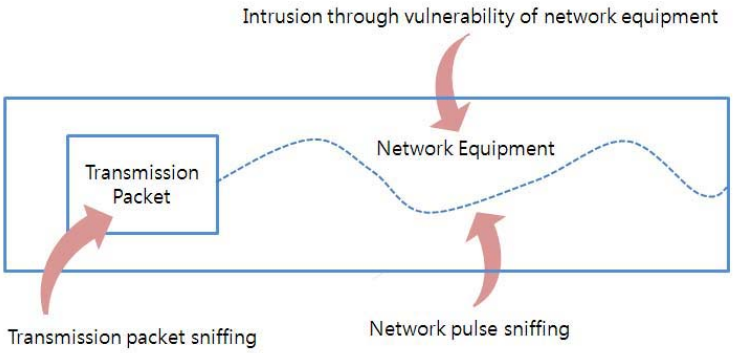


Fig. 6. Patterns of vulnerability of network

Generally, there exist the technical measures for preventing outflow of information which are classified as cut off or restriction of access to information,

encryption of data or files blocking the access made by unauthorized users, blocking file transmission or restriction to the channel of outflow, destruction of device where data or file is stored, and monitoring log in which the outflow of the data or file leaves traces. Based on vulnerability analysis about identified information asset, we executed Delphi method with professional group related to literature review and relevant field workers(3 university professor, 3 professionals working for security corporation), then we distinguished security objective from security technology and designed them as table 1. The Delphi method is that we collected opinions of professional group via survey and surveyed statistical analysis result from professional group again then repeat the collection of opinion and aggregate. This method provides a chance to modify each professionals opinion and it is positive of a chance to utilize others opinion. Currently more than 90% of technology foresight field use Delphi method and it is settled down as universal method. It has another advantage that it help get reliable assessment result via professional groups participation.

A mail and messenger securities that are to prevent a industrial technology leakage encrypt contents of e-mail and messenger via internet also filter them in observance of rules. A portable storage device security is that it implements authority control on portable storage devices(USB, mobile phone, memory card, etc) which can be connected with personal computer.

Security Objective		Security Technology	
Prevention (Protection)	Outflow Control (Responsible Use)	Mail and messenger security	
		Corporate DRM	Portable storage device security
			Document security
	Access Control (Secure Use)	DB Security	DB work monitoring and interception
			DB encryption
		Network access control	
Monitoring (Audit Trail)	Contents monitoring and filtering		

Table 1. Industrial Security Framework

A document security aiming at controlling an approach to industrial technology block an attempt to access made by unauthorized or illegal person based on encryption of the existing file. The document security also applies security regulation to the all procedures which are made from a generation of the document to disposal of the document including distribution of them. And it makes it possible to grasp a channel of the important documents outflow so that it can prevent unauthorized outflow or thief of confidential documents and product

blueprint. Database security technology consists of database activity monitoring and blocking technology. Both of technologies function as a means of protection which guards stored data in the database from unauthorized access, intentional modification and elimination of data, and contingency obstructing datas consistency. Database encryption technology not only encrypts data but also stores them. And when it is necessary, it restore the encrypted data and reads or modifies them then encrypts them again. Network access control technology protects internal network and user terminal through certain procedures that execute an isolation, cure, and permitting an access regarding terminal unmatched with security policy after inspecting a status of terminal from a stage of network access.

Consequently, contents monitoring and filtering technologies observe the distribution of industrial technology founded on a business regulation related to certain application programs. This technology also detects an inappropriate transfer of the sensible information in network.

4 Analysis of the Current Status and Limitation of Industrial Security Technology

As a result of in-depth interview research, a technology of portable storage device security is developed when various portable storage devices (secure digital card, compact flash card, memory stick, etc.) appear and new means of communication are developed. Yet there appear a problem caused by collision with controlling existing devices in interoperability.

Document security technology has restriction on program source file and a blueprint due to the big size of file, interoperability between various kinds of form of file and applications, and the needs for multi-level collaboration. And there is lack of steady state of security technology development (currently it is not possible to collect and integrate the usage history of files or the usage history of read and write. It is also impossible to control downloads and authority to use after download).

Database activity monitoring and blocking technology cannot control an access made by each user unit but can control an access made by application unit because database security technology cannot recognize which client access the database in case of access conducted through application server. When database encryption technology encrypts database, it encrypts index at the same time so that the speed of data search become slower. Also it takes long time to encrypt or decrypt large amount of data table. Unfortunately, this disadvantage may cause service halt.

Network access control technology blocks an ill-intentioned program or attempt that both of them are executed by computer users qualified for proper security level according to organizations regulation. It has emerged to develop an integrated security technology which can manage change in security policy or health condition of computer.

Currently, contents monitoring and filtering technology for ordinary corporation and public office occupy 1GB of server for 1 hour-long log of operation history and after 1 month the operation history would produce approximately 300 500GB of log. That makes it difficult to trace log after all.

5 Establishing a Strategy for Industrial Security Technology Development

As previously explained, many security technologies are being developed with various perspectives to protect industrial technology. But there is much work related to managing technologies aimed at controlling outflow and those technologies only provide protection to arranged file format. Also technologies for monitoring have a potential to commit a detection error and cannot provide real-time interception. Inconsequence, future industrial security technology is needed to be developed as policy-oriented based on organizations business process. Accordingly in this study, we deduced further technology development task as followings with professional group by Delphi method.

First, control system for different types of portable storage device conduct access control regardless of producer or operational environment and when doing data transferring to external, it still maintains access control on data from a remote computer. In detail, this control system consists of advancement of portable storage device and channel control technology, external transmission security file which supports confidentiality, integrity, and tenacity. The external transmission security file conducts encryption of document and convey decryption key to external authorized user so that user who receives security file can read relevant document without installing a certain program into terminal. The mere execution of security file let user read document under permissible range.

Industrial technology document integrated security system fulfills security and compatibility among technologies which process security related to electronic document. And it guards program source file and blueprint that possess unique feature for business process. Considering relevant work environment, security technology of program source file and blueprint should solve following security needs.

Particularly, collaboration possible industrial technology electronic document security technology should conduct an access control for user and application program at the same time. It also needs to develop integrated electronic security technology, being linked with the existing office document security technology. The current compatibility and expansion possible document security integrated technology cannot provide interoperability, when a document transmission occurs between two different organizations. So this technology prevents a document transmission in which security technology is not applied. Accordingly, API(Application Program Interface) which can control information leakage made from document distribution in the organizations should be developed.

The high-performance database security system solves vulnerability that a detour of database access through web application has and minimizes user pro-

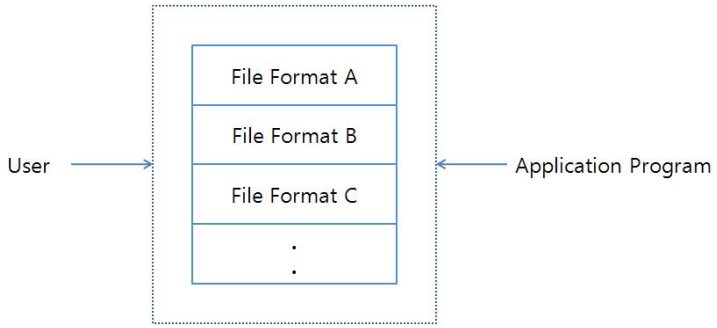


Fig. 7. Improved e Document Security

cess delay which occurs during encryption of database field. In detail, access detection and prevention technology controls non standard SQL inquiry form web application. When the trouble appears in the database security server providing connection -oriented network service, this technology guarantees accessibility allowing the application sever to access database directly.

The fast encryption(decryption) of database and search technology use encrypted index and safe key management which supports the encryption(decryption) of database field. It also provides an index search via index at the same time.

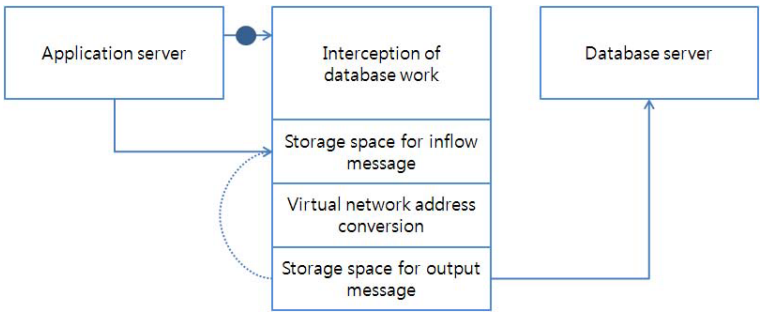


Fig. 8. Improved Database Security

Eventually, role-oriented network end point security system solves an incompatibility with remote access computer and guarantees interoperability among network access control technologies. It also supports network access control which embodies flexible industry standard infra protection and include user group and environment.

6 Exploratory Study Result regarding Industrial Security Technology

To enhance the international competitiveness by protecting up-to-date industrial technology, we have to analyze the current level of domestic industrial security technology and technical competitiveness[7]. Furthermore it is vital to establish the policy for improving the competitiveness of domestic industrial by devise a policy to support development task. In this study, we analyzed the all-pervading trend and present status of industrial security technology. Then, we conducted the deduction of national development task and analyzed current level of domestic industrial security technology for prevention of industrial technology leakage and improvement of technology.

In detail, we analyzed the status of industrial technology leakage, and grasped the main subject of leakage, channel, and method. We then designed industrial security framework with identification of industrial technology asset, research of literature, and visiting provider and demander of industrial security technology

On the next stage, we applied Delphi Method to the professional group and deduced the segmented development task. As a result, we designed the control system for different types of portable storage devices, integrated security system for industrial technology documents, high-performance database security system, and role-oriented network end point access control system.

The result of this study may be utilized to enhance an international competitive power and devise the policy for industrial security technology development ability as basic contents. Industrial security framework based on researches and practitioners is also anticipated to provide an approach method regarding industrial technology leakage prevention, detection and countermeasure. Hereafter, it is needed to develop information security management system for industrial security specialized in industrial technology protection which can carry out integrated management. There also exists necessity for further research concerning physical and managerial security system for industrial technology protection.

References

1. ISO/IEC: ISO/IEC TR 13335-4: 2000(E).: Information Technology - Guidelines for the Management of IT Security Part 4. (2000)
2. XiSEC/AEXIS Consultants.: BS7799 Information Security SME Guide. XiSEC/AEXIS Consultants. (2002)
3. Forte, Dario.: Information Security Assessment: Procedures and Methodology. Computer Fraud & Security. (2000)
4. Gartner.: Hype Cycle for Governance, Risk and Compliance Technologies. (2008)
5. Gartner.: Understanding Data Leakage. (2007)
6. Hone, Karin and Eloff, JHP.: What makes an effective information security policy?. Network security. (2002)
7. Jan Eloff, Mariki Eloff.: Information Security Management - A New Paradigm. Proceedings of SAICSIT, (2003)

8. M.M.Eloff, S.H. von Solms.: Information Security Management: An Approach to combine Process Certification And Product Evaluation. Computers & Security. (2000)
9. Dodson Rob.: Information Incident Management. Information Security Technical Report. (2001)
10. Weill, P. and M.: What IT Infrastructure Capabilities are needed to Implement e-Business Models?. Vitale MIS Quarterly Executive. (2002)

A Method to Evaluate Uncertain and Conflicting Trust and Authenticity Statements

Andreas Gutscher

Institute of Communication Networks and Computer Engineering,
Universität Stuttgart, 70569 Stuttgart, Germany
andreas.gutscher@ikr.uni-stuttgart.de *

Abstract. Countless Internet applications and platforms make it easy to communicate, to collaborate and to exchange information and opinions with a huge number of individuals. However, it is getting more and more difficult to distinguish honest and reliable individuals from malicious users distributing false information or malware. Digital signatures, webs of trust and reputation systems can help to securely identify communication partners and to estimate the trustworthiness of others, but there is a lack of trust and authenticity evaluation methods that do not show counterintuitive effects in the case of conflicting opinions.
This article proposes a new integrated method to evaluate uncertain and conflicting trust and authenticity statements. It introduces a set of operators and inference rules for combining and reasoning with these statements, it defines an approach to compute the resulting confidence values of derived statements and it compares different computation algorithms. The computation is based on a probability theoretical model in order to exclude inconsistencies and counter-intuitive effects.

1 Introduction

An increasing number of different Internet applications, platforms and social networks makes it easy to communicate with a huge number of individuals, to exchange and share information, news, photos, files and product recommendations and to socialize with people sharing similar interests. However, for users participating in a large number of social networks, discussion forums, etc. it is getting more and more difficult to find out who their new "friends" actually are and whether they can trust them.

With a *reputation system* users can share their knowledge and opinions about other users. The reputation system collects and evaluates the opinions of all users about the trustworthiness of others. On request it computes the resulting confidence value for the requested entity according to a *trust model*.

* D. Chadwick, I. You and H. Chang (Eds.): Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST2009), Purdue University, West Lafayette, USA, June 16, 2009.

The authenticity of all opinion statements should be protected, e. g., with digital signatures, to prevent manipulations and to make the evaluation verifiable to the users. Digital signatures are only useful if the users can identify the signature key holder. If no global trusted public key infrastructure is available, users can share their knowledge about the ownership of keys in a so-called *web of trust* (e. g., the PGP/GnuPG *web of trust* [1]) by exchanging digitally signed identity certificates. However, these authenticity statements are only useful if the users can verify that the issuer is trustworthy to verify the ownership of public keys. Trust and authenticity evaluation are thus highly interdependent.

Various different computational trust models, reputation systems and applications using trust have been proposed [1–10]. To obtain an intuitive and consistent trust model one must define clearly what a confidence value represents and find a sound mathematical basis for the computation with confidence values. Confidence values have strong similarities to probability values. The most sophisticated trust models are therefore based on probability theory. Maurer [4] proposed a probabilistic model in which confidence values for trust and authenticity relations correspond to probability values. However, it does not model negative opinions (distrust) and entities cannot have more than one key. Credential Networks and related models proposed by Haenni [6], Jonczy [11] and Kohlas [10] also model confidence values as probabilities. Besides degrees of support and uncertainty the confidence values can express also degrees of refutation (e. g., distrust). However, confidence values may contain either degrees of belief and ignorance[1], disbelief and ignorance, or belief and disbelief, but they cannot contain degrees of belief, disbelief and ignorance at the same time. Jøsang's Subjective Logic [5] can express opinions with degrees of belief, ignorance and disbelief (at the same time), but the approach to compute the resulting confidence values is (although based on probability theory, too) quite different. In the model of Maurer and in Credential Networks the initial confidence values are uncertain and the inference rules are deterministic, whereas in Subjective Logic the uncertainty is modeled in the operators of the inference rules, i. e., the confidence value of a conclusion is computed from the confidence values of the preconditions. Unfortunately, this leads to the problem that the resulting confidence value generally depends on the order in which the operators are applied. It seems that Subjective Logic can not be used to evaluate arbitrary networks of trust and authenticity statements without using questionable workarounds [12].

If users can express both positive (supporting) and negative (refuting) opinions, then the combination of contradictory opinions can lead to *conflicts*. In Credential Networks and Subjective Logic the probability mass associated with conflicting combinations is eliminated and the remaining probability mass is renormalized. Zadeh [13] has shown that conflict elimination and re-normalization approaches (like Dempster's rule of combination [14]) can produce counterintuitive effects.

This article proposes a new integrated approach to evaluate uncertain and conflicting trust and authenticity statements without eliminating conflict. This

[1] the terms *uncertainty* and *ignorance* are used interchangeable

avoids the counter-intuitive effects of re-normalizations. The trust model is based on a combination of the inference rules from [15] and the calculus and operators from [16], extended by a new operator for reasoning with authenticity statements. Sect. 2 describes the representation of trust and authenticity statements, Sect. 3 the representation of confidence values and the corresponding operators. The new inference rules are formulated in Sect. 4. Sect. 5 proposes different algorithms to compute the resulting confidence value, Sect. 6 compares the computation time of these algorithms and Sect. 7 concludes the article.

2 Model of Trust and Authenticity Statements

An *opinion* refers to a trust or authenticity statement H_j with an associated confidence value t_j. A *first-hand* opinion is an opinion that is based only on the experience and knowledge of a *single* entity (the *trustor* or *issuer*) and that is *independent* of other opinions. A *second-hand* opinion is an opinion that is derived from other opinions and that is thus not independent.

We define *trust* as *"a unidirectional relation between a trustor and a trustee expressing the strong belief of the trustor that the trustee will behave as expected with respect to a particular capability within a particular context"* [15]. Therefore we represent the standard form of a trust statement as follows:

$$\mathrm{Trust}(\mathit{trustor}, \mathit{trustee}, r, h_{\min}..h_{\max}) \tag{1}$$

The *trustor* can be an entity or a key, the *trustee* an entity, a description or a key (see Tab. 1). An *entity* (E_A, E_B, ...) can be a person, an organization, a network node, etc. referred to by a *local* identifier. To exchange opinions with others users have to use unique *descriptions* or *public keys* to refer to other entities. A *description* (D_A, D_B, ...) consists of a list of names, identifiers or attributes that uniquely identifies the described entity. Entities may have several different descriptions. A *public key* (K_A, K_B, ...) is the public part of an asymmetric key pair. The holder uses the key pair to sign trust or authenticity statements (certificates). An entity can use several different key pairs at the same time.

Table 1. Trust and authenticity statements (relations and certificates)

	Trust statements		Authenticity
	Standard form	Internal form	statements
Relation	$\mathrm{Trust}(E_A, E_B, r, h_{\min}..h_{\max})$	$\mathrm{Trust}(E_A, E_B, r, h, l)$	$\mathrm{Auth}(E_A, K_B, E_B)$
	$\mathrm{Trust}(E_A, K_B, r, h_{\min}..h_{\max})$	$\mathrm{Trust}(E_A, K_B, r, h, l)$	$\mathrm{Auth}(E_A, D_B, E_B)$
	$\mathrm{Trust}(E_A, D_B, r, h_{\min}..h_{\max})$	$\mathrm{Trust}(E_A, D_B, r, h, l)$	$\mathrm{Auth}(E_A, K_B, D_B)$
Certificate	$\mathrm{Trust}(K_A, K_B, r, h_{\min}..h_{\max})$	$\mathrm{Trust}(K_A, K_B, r, h, l)$	$\mathrm{Auth}(K_A, K_B, D_B)$
	$\mathrm{Trust}(K_A, D_B, r, h_{\min}..h_{\max})$	$\mathrm{Trust}(K_A, D_B, r, h, l)$	

The capability r refers to an application specific capability (r_1, r_2, ...) or to the capability r_{PKI}, which represents the capability to honestly and carefully verify that a description uniquely refers to the holder of a particular key pair.

We distinguish different types of trust identified by a different number of recommendation hops (h): *Functional trust* expresses the belief that the trustee *has* the capability r and is described by $h = 0$. *Recommendation trust* for $h = 1$ hop expresses the belief that the trustee can *recommend* someone with capability r, *recommendation trust* for $h = 2$ hops that the trustee can *recommend someone who can recommend* someone with capability r, etc. Each standard form trust statement can specify the desired range of recommendation hops $h_{\min}..h_{\max}$.

For the evaluation of trust statements we need in addition trust statements in the slightly different *internal form*. These trust statements refer not to a range, but to a single recommendation hop value $h \geq 0$ and they have an additional parameter, the *chain length* $l \geq 1$:

$$\text{Trust}(\mathit{trustor}, \mathit{trustee}, r, h, l) \tag{2}$$

Trust is not transitive in general, but trust statements can be combined in certain cases to trust chains according to the transitive trust inference rule (7) described in Sect. 4.1. The chain length l of the derived trust statement refers to the number of first-hand trust statements in the trust chain.

Authenticity statements express the strong belief of the issuer that a description belongs to an entity, that a public key belongs to an entity or that a description belongs to the holder of a public key:

$$\text{Auth}(\mathit{issuer}, \mathit{actor}_1, \mathit{actor}_2) \tag{3}$$

The *issuer* is an entity or a public key, actor_1 and actor_2 are entities, descriptions or public keys. All four possible combinations are listed in Tab. 1[2].

3 Confidence Values

This section introduces discrete and continuous confidence values as well as operators for reasoning with discrete confidence values. Users express their opinions with continuous confidence values while the discrete confidence values are used internally only for reasoning with opinions.

3.1 Representation of Discrete and Continuous Confidence Values

Users can have different and possibly conflicting opinions about trust and authenticity statements. Therefore, we can not definitively decide whether a statement H is "true" or "false". We can only evaluate known indications that *support* or *refute* H. It is possible that neither supporting nor refuting or that both supporting and refuting indications for H are found. Therefore we describe knowledge of supporting and refuting indications *independently*. For each statement H we introduce the *propositions* H^+ and H^- to describe that the reputation

[2] certificates can not contain local identifiers for entities ($E_A, E_B, \ldots$) because they would be meaningless to other entities

system is aware of indications that imply that H must be true and that H must be false, respectively. We also introduce the four *discrete* confidence values *belief* $(+)$, *ignorance* $(\varnothing)$, *disbelief* $(-)$ and *conflict* $(\pm)$ to represent the four possible combinations of these propositions (see Tab. 2). They can be seen as "truth values" of a paraconsistent logic [16].

Table 2. Discrete confidence values

Propositions	Discrete confidence value	Semantics
$\{H^+\}$	$t' = +$ (*belief*)	"the indications imply that H must be true"
$\{\}$	$t' = \varnothing$ (*ignorance*)	"there are no relevant indications about H"
$\{H^-\}$	$t' = -$ (*disbelief*)	"the indications imply that H must be false"
$\{H^+, H^-\}$	$t' = \pm$ (*conflict*)	"the indications imply that H must be true and that H must be false at the same time"

As statements can in fact not be both true and false at the same time we can conclude that *first-hand* opinions can not have the confidence value *conflict*. However, if we combine statements of *different* (disagreeing) entities, it is possible to find both H^+ and H^-, i. e., the confidence value of derived (*second-hand*) opinions can be *conflict*. Conflict must not be confused with partial support and partial refutation (*ambivalent opinions*). An entity that has for example experienced some positive and some negative interactions can express this opinion with *continuous* confidence values.

Continuous confidence values $t = (b, i, d, c)$ with $b, i, d, c \in [0, 1]$ and $b + i + d + c = 1$ express *degrees* of belief, ignorance, disbelief and conflict. The value b represents the issuer's subjective estimation of the probability that there are indications supporting (but no refuting) H. Similarly, d represents the subjective estimation of the probability that there are indications refuting (but no supporting) H. c represents the subjective estimation of the probability that there are both supporting and refuting indications for H at the same time, and i represents the subjective estimation of the probability that there are neither supporting nor refuting indications for H. For the same reason as before, c must be zero in all first-hand opinions, whereas second-hand opinions can contain conflict. Nevertheless, ambivalent first-hand opinions can be expressed by continuous confidence values with both $b > 0$ and $d > 0$. A user that has made many positive and few negative experiences can choose, for example, a first-hand confidence value with $b = 0.7$ and $d = 0.1$ (i. e., $t = (0.7, 0.2, 0.1, 0)$. Thus, in first-hand statements b can be seen as the lower bound and $1 - d$ as the upper bound for the estimated subjective probability that H must be true.

The degrees of ignorance and conflict in resulting confidence values have different meanings, and applications should handle high degrees of ignorance and conflict differently: A high degree of ignorance indicates that the reputation system has little information about the requested statement and suggests searching more relevant statements, if possible. A high degree of conflict, however, shows that the requested statement H is controversial. This suggests that the requester

should verify whether the trust and authenticity assignments he made and that cause the conflict are correct.

Continuous confidence value can be condensed to a single value w, if desired:

$$w = b + w_i i + w_d d + w_c c \tag{4}$$

The parameters w_i, w_d and w_c represent weights for the degrees of ignorance, disbelief and conflict, e. g., $w_i = 0.5$, $w_d = -1$ and $w_c = 0$. They can be chosen according to the preferences of the application and allow for rather optimistic or rather pessimistic behavior in the cases of uncertainty and conflict.

3.2 Operators to Combine Discrete Confidence Values

This section describes the recommendation and authentication operators. The operators define whether H_z^+ and H_z^- can be derived from a set of premises (H_x^+, H_x^-, H_y^+, H_y^-). The short notation with statements is provided for convenience and will be used to formulate the inference rules in Sect. 4.

Recommendation Operator The recommendation operator ($\otimes$) is used to concatenate two trust statements or a trust with an authenticity statement. It is reasonable for a user to adopt the opinions of trustworthy entities. However, it is not reasonable (it is in fact even dangerous) to assume that untrustworthy (malicious or incompetent) entities always tell the opposite of the truth. Instead, opinions of untrustworthy entities should be ignored. Therefore, we do not draw any conclusions from H_x^-. The operator is thus defined as follows:

$$\frac{H_x \otimes H_y}{H_z} \Leftrightarrow \frac{H_x^+ \quad H_y^+}{H_z^+}, \frac{H_x^+ \quad H_y^-}{H_z^-} \tag{5}$$

This reads as follows: H_z follows from a combination of H_x and H_y with the recommendation operator. If there are supporting indications for H_x and for H_y, then infer H_z^+. If there are supporting indications for H_x and refuting indications for H_y, then infer H_z^-. Fig. 1 (left) shows the corresponding "truth table".

$t'_z = t'_x \otimes t'_y$		t'_x			
		$+$	$\varnothing$	$-$	$\pm$
t'_y	$+$	$+$	$\varnothing$	$\varnothing$	$+$
	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
	$-$	$-$	$\varnothing$	$\varnothing$	$-$
	$\pm$	$\pm$	$\varnothing$	$\varnothing$	$\pm$

$\odot$	$+$	$\varnothing$	$-$	$\pm$
$+$	$+$	$\varnothing$	$-$	$\pm$
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$-$	$-$	$\varnothing$	$\varnothing$	$-$
$\pm$	$\pm$	$\varnothing$	$-$	$\pm$

Fig. 1. Recommendation and authentication operator truth tables

Authentication Operator The authentication operator ($\odot$) is used to reason with two authenticity relations between entities, descriptions and public keys:

$$\frac{H_x \odot H_y}{H_z} \Leftrightarrow \frac{H_x^+ \quad H_y^+}{H_z^+}, \frac{H_x^+ \quad H_y^-}{H_z^-}, \frac{H_x^- \quad H_y^+}{H_z^-} \tag{6}$$

The operator definition can be understood as follows: Assume H_x and H_y represent statements like "A and B belong together" and "B and C belong together", respectively. If we have supporting indications for both statements, then this supports that A and C belong together (H_z). If we have indications that A and B belong together but that B does not belong to C, then we conclude that A does not belong to C either. If neither A belongs to B nor does B belong to C, then we can draw no conclusion about A and C. Fig. 1 (right) shows the corresponding truth table.

4 Inference Rules

The inference rules specify which conclusions the reputation system can draw from a set of given trust and authenticity propositions.

4.1 Transitive Trust Inference Rule

This inference rule describes the *transitivity* property of trust statements. It defines in which cases two trust statements for the same capability r can be combined with the recommendation operator in order to derive a new trust statement from the trustor of the first statement (A) to the trustee of the second statement (C). The trustor A can be an entity (E_A) or a public key (K_A). The second statement can be a trust statement or a trust certificate, i. e., B can be an entity (E_B) or a public key (K_B). The final trustee C can be an entity (E_C), a public key (K_C) or a description (D_C).

$$\frac{\text{Trust}(A, B, r, h + l_2, l_1) \otimes \text{Trust}(B, C, r, h, l_2)}{\text{Trust}(A, C, r, h, l_1 + l_2)} \tag{7}$$

This inference rule differs from other proposed transitive trust inference rules in that it allows the combination of trust statements only if the number of recommendation hops matches: The number of recommendation hops of the first statement must equal the sum of the recommendation hops plus the chain length of the second statement. The chain length of the resulting statement is the sum of the chain lengths of the input statements. This ensures that the recommendation hop value of the trust statements decreases by one throughout the chain of first-hand trust relations (e. g., $h = 2$, $h = 1$, $h = 0$).

The example in Fig. 2 illustrates the inference rule. The transitive trust inference rule allows to combine $H_1^+ = \text{Trust}^+(E_A, E_B, r, 2, 1)$ with $H_2^+ = \text{Trust}^+(E_B, E_C, r, 1, 1)$ to $H_4^+ = \text{Trust}^+(E_A, E_C, r, 1, 2)$ and then H_4^+ with $H_3^- = \text{Trust}^-(E_C, E_D, r, 0, 1)$ to $H_5^- = \text{Trust}^-(E_A, E_D, r, 0, 3)$.

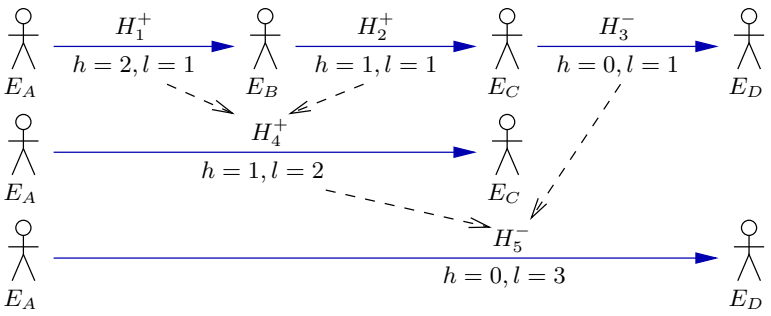


Fig. 2. Example for application of the transitive trust inference rule

4.2 Trust in Entities, Keys and Descriptions

A number of simple rules allow to infer from trust assigned to an entity to trust assigned to the holder of a key and to trust assigned to an entity identified by a description, and vice versa. If an entity is trustworthy, then the holder of a key that belongs to this entity is trustworthy, too, and vice versa:

$$\frac{\text{Auth}(E_A, K_C, E_C) \otimes \text{Trust}(E_A, E_C, r, h, l)}{\text{Trust}(E_A, K_C, r, h, l)} \tag{8}$$

$$\frac{\text{Auth}(E_A, K_C, E_C) \otimes \text{Trust}(E_A, K_C, r, h, l)}{\text{Trust}(E_A, E_C, r, h, l)} \tag{9}$$

If an entity is trustworthy, then the entity identified by a description that belongs to this entity is trustworthy, too, and vice versa:

$$\frac{\text{Auth}(E_A, D_C, E_C) \otimes \text{Trust}(E_A, E_C, r, h, l)}{\text{Trust}(E_A, D_C, r, h, l)} \tag{10}$$

$$\frac{\text{Auth}(E_A, D_C, E_C) \otimes \text{Trust}(E_A, D_C, r, h, l)}{\text{Trust}(E_A, E_C, r, h, l)} \tag{11}$$

If the holder of a key is trustworthy, then the entity identified by a description that belongs to this key holder is trustworthy, too, and vice versa. This applies to trust relations and trust certificates:

$$\frac{\text{Auth}(E_A, K_C, D_C) \otimes \text{Trust}(E_A, K_C, r, h, l)}{\text{Trust}(E_A, D_C, r, h, l)} \tag{12}$$

$$\frac{\text{Auth}(E_A, K_C, D_C) \otimes \text{Trust}(E_A, D_C, r, h, l)}{\text{Trust}(E_A, K_C, r, h, l)} \tag{13}$$

$$\frac{\text{Auth}(K_A, K_C, D_C) \otimes \text{Trust}(K_A, K_C, r, h, l)}{\text{Trust}(K_A, D_C, r, h, l)} \tag{14}$$

$$\frac{\text{Auth}(K_A, K_C, D_C) \otimes \text{Trust}(K_A, D_C, r, h, l)}{\text{Trust}(K_A, K_C, r, h, l)} \tag{15}$$

4.3 Local Authenticity Inference Rule

If an entity E_A has *partial* knowledge about whether an entity E_B is the holder of a key K_B, whether a description D_B refers to the entity E_B or whether the description D_B refers to the holder of the key K_B, then it can draw further conclusions about the confidence values of the authenticity statements between E_B, K_B and D_B. If the confidence values of two corresponding authenticity relations are known, then the confidence value of the third authenticity relation can be derived with the authentication operator:

$$\frac{\mathrm{Auth}(E_A, K_C, D_C) \odot \mathrm{Auth}(E_A, K_C, E_C)}{\mathrm{Auth}(E_A, D_C, E_C)} \tag{16}$$

$$\frac{\mathrm{Auth}(E_A, K_C, D_C) \odot \mathrm{Auth}(E_A, D_C, E_C)}{\mathrm{Auth}(E_A, K_C, E_C)} \tag{17}$$

$$\frac{\mathrm{Auth}(E_A, K_C, E_C) \odot \mathrm{Auth}(E_A, D_C, E_C)}{\mathrm{Auth}(E_A, K_C, D_C)} \tag{18}$$

4.4 Authenticity Inference with Authenticity Confirmation

If a trustor (E_A or K_A) trusts a trustee (E_B or K_B) to issue only correct authenticity relations or identity certificates (property r_{PKI}), then the trustor can conclude that authenticity relations or identity certificates of the trustee are correct:

$$\frac{\mathrm{Trust}(E_A, E_B, r_{\mathrm{PKI}}, 0, l) \otimes \mathrm{Auth}(E_B, K_C, D_C)}{\mathrm{Auth}(E_A, K_C, D_C)} \tag{19}$$

$$\frac{\mathrm{Trust}(E_A, K_B, r_{\mathrm{PKI}}, 0, l) \otimes \mathrm{Auth}(K_B, K_C, D_C)}{\mathrm{Auth}(E_A, K_C, D_C)} \tag{20}$$

$$\frac{\mathrm{Trust}(K_A, K_B, r_{\mathrm{PKI}}, 0, l) \otimes \mathrm{Auth}(K_B, K_C, D_C)}{\mathrm{Auth}(K_A, K_C, D_C)} \tag{21}$$

4.5 Uniqueness Conditions

Two further conclusions can be drawn from the condition that each public key has only one holder and that each description refers to only one entity. If A knows that E_B is the holder of K_B, then it can infer that all other entities are not the holder of K_B. Similarly, if A knows that E_B has the description D_B, then it can infer that all other entities do not have the description D_B (A can be an entity or a key).

$$\frac{Auth^{+}(A, K_B, E_B)}{Auth^{-}(A, K_B, E_j)}, \frac{Auth^{+}(A, D_B, E_B)}{Auth^{-}(A, D_B, E_j)} \qquad \forall E_j \neq E_B \tag{22}$$

5 Confidence Value Computation

The reputation system collects all issued first-hand trust and authenticity opinions H_j with associated continuous confidence value t_j (with $c_j = 0$). Users can then send requests in the form of a standard form trust statement or an authenticity statement to the reputation system. The reputation system then processes all collected opinions. It applies the inference rules to derive trust and authenticity statements and it computes the resulting continuous confidence value t_0 of the requested statement H_0 from the confidence values of the relevant first-hand statements. As the components of the continuous first-hand confidence values (b, i and d) represent probabilities, we define the resulting confidence value by a random experiment and propose different algorithms for the computation of the resulting confidence value.

5.1 Probabilistic Model for the Confidence Value Computation

The components of the computed *resulting* confidence value $t_0 = (b_0, i_0, d_0, c_0)$ for H_0 are computed from the combination of all available first-hand opinions with the inference rules under the assumption that the confidence values of the opinions of the requestor are correct. In short, b_0 is the computed lower bound for the probability that the combination of the available first-hand opinions leads to the conclusion that H_0 must be true (but not that H_0 must be false). Similarly, d_0 is the computed lower bound for the probability that the combination of the available first-hand opinions leads to the conclusion that H_0 must be false (but not that H_0 must be true). The degree of conflict c_0 is the computed probability that the combination of the first-hand opinions leads to the contradicting conclusion that H_0 must be both true *and* false at the same time. The degree of ignorance is the remaining probability $i_0 = 1 - b_0 - d_0 - c_0$.

The following description of a random experiment provides a more detailed definition for t_0: We assume that the reputation system has collected J first-hand opinions, i. e., the statements H_j ($j = 1, 2, \ldots J$) with associated continuous confidence values $t_j = (b_j, i_j, d_j, 0)$. For each first-hand statement H_j choose a *discrete* confidence value t'_j from $\{+, \varnothing, -\}$ according to the weights b_j, i_j and d_j, i. e., choose $t'_j = +$ with probability b_j, $t'_j = \varnothing$ with probability i_j and $t'_j = -$ with probability d_j. Statements with the discrete confidence value *ignorance* don't contribute knowledge and can be discarded[3]. Each remaining first-hand statement H_j with associated discrete confidence value t'_j corresponds to a set of first-hand propositions according to Tab. 2.

The inference rules always operate on trust propositions in the *internal* representation. We therefore have to replace each standard-form trust statement $\mathrm{Trust}(A, B, r, h_{\min}..h_{\max})$ by a list of single-hop trust statements in *internal form* with chain length $l = 1$: $\mathrm{Trust}(A, B, r, h_{\min}, l)$, $\mathrm{Trust}(A, B, r, h_{\min} + 1, l)$, $\ldots$, $\mathrm{Trust}(A, B, r, h_{\max}, l)$. The internal trust statements inherit their assigned

[3] this optimization does not change the resulting confidence value, the resulting continuous confidence value t_0 nevertheless contains the correct degree of ignorance

discrete confidence value from the standard-form trust statement. Next, we apply all inference rules (see Sect. 4) to derive all (positive and negative) deducible propositions from the set of all known first-hand propositions and all already derived propositions. To get back to trust statements in standard form we conclude $H_0^+ = \mathrm{Trust}^+(A, B, r, h_{\min}..h_{\max})$ if we have been able to derive a proposition $H_{0,h}^+ = \mathrm{Trust}^+(A, B, r, h, l)$ with $h_{\min} \leq h \leq h_{\max}$. Similarly, we conclude H_0^- if we have been able to derive a proposition $H_{0,h}^-$.

To obtain the resulting continuous confidence value of a requested trust or authenticity statement we compute the probability that the random experiment leads to a set of first-hand propositions from which we can derive positive and negative propositions for the requested statement H_0. The components of the resulting confidence value $t_0 = (b_0, i_0, d_0, c_0)$ are defined as follows: b_0 is the probability that H_0^+ (but not H_0^-) can be derived and d_0 is the probability that H_0^- (but not H_0^+) can be derived. The probability that neither H_0^+ nor H_0^- can be derived is i_0, and c_0 is the probability that both H_0^+ and H_0^- can be derived.

In contrast to other trust models (e. g., [5, 6, 10, 11]) we propose *not to eliminate* the degree of conflict, not only to avoid counter-intuitive effects of renormalizations but also because it provides valuable information to the requesting user or application (see Sect. 3.1).

5.2 Approximation and Exact Computation Algorithms

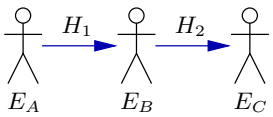
This section presents different possibilities to implement the computation of an approximation or of the exact value of the resulting continuous confidence value t_0 according to Sect. 5.1. All exact algorithms return the same resulting confidence value t_0, but differ in computation time. The result of the approximation gets arbitrarily close to the exact result if the number of iterations is sufficiently large.

To keep the computation time small we recommend for all algorithms to *precompute all possible paths*: We first set up a "superposition" of possible first-hand propositions. For each statement H_j with continuous confidence value $t_j = (b_j, i_j, d_j, 0)$ we select H_j^+ if $b_j > 0$ and we select (possibly in addition) H_j^- if $d_j > 0$. Then we translate all trust propositions into the internal form, apply all inference rules and record the dependencies, i. e., we trace which sets of first-hand propositions (premises) allow to derive which conclusions. Each set of first-hand propositions that allows to (directly or indirectly) derive the positive requested proposition H_0^+ is called a *positive path* for H_0, each set that allows to derive the negative proposition H_0^- a *negative path* for H_0. Next, we select the set of positive paths and the set of negative paths for H_0 and minimize these paths, i. e., we remove all paths that contain at least one other path in the set. We finally obtain the set of minimal positive paths $A^+ = \{a_1^+, a_2^+, \ldots a_{k^+}^+\}$ and the set of minimal negative paths $A^- = \{a_1^-, a_2^-, \ldots a_{k^-}^-\}$.

Approximation with Monte-Carlo Simulation An obvious approach to determine an *approximation* for the resulting confidence value is to run the

described random experiment N times and to count in how many experiments the selected set of first-hand propositions contains at least one positive (but no negative) path (n_b), no paths (n_i), at least one negative (but no positive) path (n_d) or both positive and negative paths (n_c). The approximation for the confidence value is $\bar{t}_0 = \frac{1}{N}(n_b, n_i, n_d, n_c)$. The choice of N allows to adjust the trade-off between precision and computation time.

Possible Worlds Algorithm An simple algorithm to compute the exact value is to go through the list of all possible combinations of first-hand propositions (so-called *possible worlds*), to compute the probability of each of those possible worlds and to check for each world whether the set of first-hand propositions of this world contains the minimal paths. The sum of all probabilities of all worlds that contain at least one positive and at least one negative path is c_0, b_0 is the sum of probabilities of all worlds that contain at least one positive, but no negative path, and d_0 the sum of probabilities of all worlds that contain at least one negative, but no positive path. The degree of ignorance is $i_0 = 1 - b_0 - d_0 - c_0$.



j	H_j	b_j	i_j	d_j
1	Trust$(E_A, E_B, r_1, 1)$	0.5	0.15	0.35
2	Trust$(E_B, E_C, r_1, 0)$	0.7	0.1	0.2

$H_{1,1}$	$H_{2,0}$	Probability	H_0
$H^+_{1,1}$	$H^+_{2,0}$	$b_1 b_2$	H^+_0
$H^+_{1,1}$		$b_1 i_2$	
$H^+_{1,1}$	$H^-_{2,0}$	$b_1 d_2$	H^-_0
	$H^+_{2,0}$	$i_1 b_2$	
		$i_1 i_2$	
	$H^-_{2,0}$	$i_1 d_2$	
$H^-_{1,1}$	$H^+_{2,0}$	$d_1 b_2$	
$H^-_{1,1}$		$d_1 i_2$	
$H^-_{1,1}$	$H^-_{2,0}$	$d_1 d_2$	

Fig. 3. Scenario and possible worlds table of Example 1

Example 1: Simple Trust Chain with Possible Worlds Algorithm The example scenario in Fig. 3 (left) consists of two trust statements: H_1 is a recommendation trust statement for one recommendation hop ($h = 1$), and H_2 is a functional trust statement ($h = 0$). E_A wants to compute the resulting functional trustworthiness of E_C ($H_0 = \text{Trust}(E_A, E_C, r_1, 0)$).

First, the trust statements in standard form have to be replaced by corresponding trust statements in internal form: H_1 by $H_{1,1} = \text{Trust}(E_A, E_B, r_1, 1, 1)$ and H_2 by $H_{2,0} = \text{Trust}(E_B, E_C, r_1, 0, 1)$. Both refer to the same property r_1, it is therefore possible to combine $H_{1,1}$ and $H_{2,0}$ with the transitive trust inference rule (7) to the new functional trust statement $H_{0,0} = \text{Trust}(E_A, E_C, r_1, 0, 2)$: $H^+_{1,1}, H^+_{2,0} \vdash H^+_{0,0}$ ($H^+_{1,1}$ and $H^+_{2,0}$ allow to drive $H^+_{0,0}$) and $H^+_{1,1}, H^-_{2,0} \vdash H^-_{0,0}$. Thus, there is only one positive path $a^+_1 = \{H^+_{1,1}, H^+_{2,0}\}$ and one negative path $a^-_1 = \{H^+_{1,1}, H^-_{2,0}\}$ for $H_{0,0}$ and thus for H_0.

Fig. 3 (right) shows all possible combinations of the first-hand propositions, the probability that this world occurs and the propositions that can be derived in this world. There are no worlds in which both H_0^+ and H_0^- can be derived, thus $c_0 = 0$. H_0^+ can be derived only in the first world, therefore $b_0 = b_1 b_2$. Similarly, H_0^- can be derived only in the third world, therefore $d_0 = b_1 d_2$. The degree of ignorance is the remaining probability mass $i_0 = 1 - b_0 - d_0 - c_0$. With the values in Fig. 3 we obtain $t_0 = (0.35, 0.55, 0.1, 0)$.

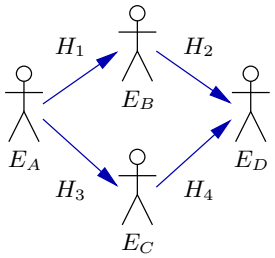
Grouped Possible Worlds Algorithm The possible worlds algorithm can be improved by subdividing the set of relevant first-hand statements into as few groups $g_1, \ldots g_u$ as possible. Two statements, H_j and H_m, belong to the same group if and only if the following condition holds for each positive, negative and conflicting path: If the path contains a proposition for H_j (H_j^+ or H_j^-), then it must also contain a proposition for H_m (H_m^+ or H_m^-).

In the preparation step we construct for each group a list of all relevant combinations of propositions of the statements in the group. This list contains all combinations that contain exactly one proposition (i. e., either H_j^+ or H_j^-)[4]for each statement and that is identical to the corresponding section of at least one (positive or negative) path. An additional element of this list consists of an empty set. It represents all remaining possible combinations of propositions, i. e., all combinations that contain neither H_j^+ nor H_j^- for at least one statement H_j of the group. We can subsume these combinations because the have the same effect on the derivability of propositions of H_0. For each element of the list we compute the probability that this combination will occur (within the group) from the continuous confidence values of the statements. The probability associated with the empty set is the sum of the probabilities of the contained combinations of propositions (i. e., the remaining probability). Thus, the sum of all probabilities is one.

Next, in the main step, we go through all possible worlds. Each world consists of one possible combination of these prepared proposition-combinations of the groups, i. e., for each groups we select one proposition-combination from the prepared list of the group. We multiply the precomputed probabilities of the chosen proposition-combinations to obtain the resulting probability of the world. Finally, we compute b_0, i_0, d_0 and c_0 just as in the possible worlds algorithm.

Example 2: Parallel Trust Chain with Grouped Possible Worlds Algorithm The scenario in Fig. 4 consists of two parallel trust chains from E_A to E_D. E_A requests the confidence value for the resulting functional trustworthiness of E_D ($H_0 = \text{Trust}(E_A, E_D, r_1, 0)$). The trust statements in standard form are replaced by statements in internal form: H_1 by $H_{1,1} = \text{Trust}(E_A, E_B, r_1, 1, 1)$, H_2 by $H_{2,0} = \text{Trust}(E_B, E_D, r_1, 0, 1)$, H_3 by $H_{3,1} = \text{Trust}(E_A, E_C, r_1, 1, 1)$ and H_4 by $H_{4,0} = \text{Trust}(E_C, E_D, r_1, 0, 1)$. We can combine $H_{1,1}$ with $H_{2,0}$ and $H_{3,1}$ with $H_{4,0}$ with the transitive trust inference rule (7). We obtain

[4] no combination can contain both H_j^+ and H_j^- because $c_j = 0$



j	H_j	b_j	i_j	d_j
1	Trust$(E_A, E_B, r_1, 1)$	0.8	0.15	0.05
2	Trust$(E_B, E_D, r_1, 0)$	0.7	0.1	0.2
3	Trust$(E_A, E_C, r_1, 1)$	0.9	0.1	0
4	Trust$(E_C, E_D, r_1, 0)$	0.8	0.1	0.1

Fig. 4. Scenario of Example 2

two positive paths $A^+ = \{\{H^+_{1,1}, H^+_{2,0}\}, \{H^+_{3,1}, H^+_{4,0}\}\}$ and two negative paths $A^- = \{\{H^+_{1,1}, H^-_{2,0}\}, \{H^+_{3,1}, H^-_{4,0}\}\}$. Propositions for $H_{1,1}$ and $H_{2,0}$ appear always together in paths, the same holds for $H_{3,1}$ and $H_{4,0}$. Therefore we can divide the statements into two groups $g_1 = \{H_{1,1}, H_{2,0}\}$ and $g_1 = \{H_{3,1}, H_{4,0}\}$.

In the preparation step we set up a list for each group that contains all relevant possible combinations of the propositions and their probabilities (see Tab. 3). For each group we find three relevant combinations: one combination supports a positive path and one a negative path. The third entry with the empty set represents the remaining combinations.

Table 3. Preparation step for the groups in Example 2

Propositions g_1	Probability
$\{H^+_{1,1}, H^+_{2,0}\}$	$b_1 b_2$
$\{H^+_{1,1}, H^-_{2,0}\}$	$b_1 d_2$
$\{\}$	$1 - b_1 b_2 - b_1 d_2$

Propositions g_2	Probability
$\{H^+_{3,1}, H^+_{4,0}\}$	$b_3 b_4$
$\{H^+_{3,1}, H^-_{4,0}\}$	$b_3 d_4$
$\{\}$	$1 - b_3 b_4 - b_3 d_4$

Table 4. Confidence value computation in the parallel trust chain example

Table 5. Confidence value computation in Example 2

g_1	g_2	Probability	H_0
$\{H^+_{1,1}, H^+_{2,0}\}$	$\{H^+_{3,1}, H^+_{4,0}\}$	$b_1 b_2 b_3 b_4$	H^+_0
$\{H^+_{1,1}, H^+_{2,0}\}$	$\{H^+_{3,1}, H^-_{4,0}\}$	$b_1 b_2 b_3 d_4$	H^+_0, H^-_0
$\{H^+_{1,1}, H^+_{2,0}\}$	$\{\}$	$b_1 b_2 (1 - b_3 b_4 - b_3 d_4)$	H^+_0
$\{H^+_{1,1}, H^-_{2,0}\}$	$\{H^+_{3,1}, H^+_{4,0}\}$	$b_1 d_2 b_3 b_4$	H^+_0, H^-_0
$\{H^+_{1,1}, H^-_{2,0}\}$	$\{H^+_{3,1}, H^-_{4,0}\}$	$b_1 d_2 b_3 d_4$	H^-_0
$\{H^+_{1,1}, H^-_{2,0}\}$	$\{\}$	$b_1 d_2 (1 - b_3 b_4 - b_3 d_4)$	H^-_0
$\{\}$	$\{H^+_{3,1}, H^+_{4,0}\}$	$(1 - b_1 b_2 - b_1 d_2) b_3 b_4$	H^+_0
$\{\}$	$\{H^+_{3,1}, H^-_{4,0}\}$	$(1 - b_1 b_2 - b_1 d_2) b_3 d_4$	H^-_0
$\{\}$	$\{\}$	$(1 - b_1 b_2 - b_1 d_2)(1 - b_3 b_4 - b_3 d_4)$	

To compute the resulting confidence value t_0 for H_0 we set up Tab. 5 with all $3 \cdot 3 = 9$ possible combinations of the entries in the lists (possible worlds), the probabilities of each world and the derivable propositions for H_0. Then we add the probabilities and obtain $b_0 = b_1b_2b_3b_4 + b_1b_2(1 - b_3b_4 - b_3d_4) + (1 - b_1b_2 - b_1d_2)b_3b_4$, $i_0 = (1-b_1b_2-b_1d_2)(1-b_3b_4-b_3d_4)$, $d_0 = b_1d_2b_3d_4+b_1d_2(1-b_3b_4-b_3d_4) + (1 - b_1b_2 - b_1d_2)b_3d_4$ and $c_0 = b_1b_2b_3d_4 + b_1d_2b_3b_4$. With the values in Fig. 4 we obtain $t_0 = (0.7112, 0.0532, 0.07, 0.1656)$.

Computation with Inclusion-exclusion Formula This algorithm computes the exact resulting confidence value directly from the minimal positive and negative paths for H_0. In addition, we need the set of minimal *conflicting paths*. Therefore we set up all possible combinations consisting of one positive and one negative path ($a_x^{\pm} = a_y^{+} \cup a_z^{-}$ with $y = 1, \dots k^{+}$, $z = 1, \dots k^{-}$), minimize the set and obtain $A^{\pm} = \{a_1^{\pm}, a_2^{\pm}, \dots a_{k^{\pm}}^{\pm}\}$ (with $k^{\pm} \leq k^{+}k^{-}$). A useful optimization is to eliminate all paths that contain both H_j^{+} and H_j^{-} (because $c_j = 0$).

First, we compute the degree of conflict c_0 from the confidence values of the first-hand statements in the set of minimal paths with the inclusion-exclusion-formula ($I(A)$): c_0 is the probability that a possible world chosen according to Sect. 5.1 will contain at least one conflicting path. Thus, we add the probabilities of all minimal paths, subtract the probabilities of all unions of two minimal paths, add the probabilities of all unions of three minimal paths, etc.:

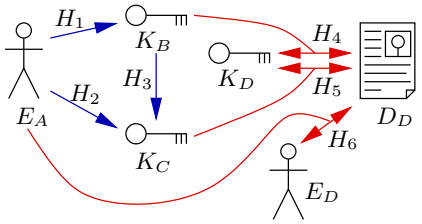
$$\begin{aligned} c_0 = & I(A^{\pm}) = \sum_{n=1}^{k^{\pm}} (-1)^{n+1} \sum_{1 \leq j_1 < \dots < j_n \leq k^{\pm}} P(a_{j_1}^{\pm} \cup \dots \cup a_{j_n}^{\pm}) \\ & = \sum_{j_1=1}^{k^{\pm}} P(a_{j_1}^{\pm}) - \sum_{1 \leq j_1 < j_2 \leq k^{\pm}} P(a_{j_1}^{\pm} \cup a_{j_2}^{\pm}) + \dots + (-1)^{k^{\pm}+1} P(a_1^{\pm} \cup \dots \cup a_{k^{\pm}}^{\pm}) \end{aligned} \tag{23}$$

$P(a)$ denotes the probability that path a is contained in the set of first-hand propositions of a chosen possible world:

$$P(a) = \prod_{j: H_j^{+} \in a \text{ or } H_j^{-} \in a} p_j \qquad \text{with } p_j = \begin{cases} 0 & \text{if } H_j^{+} \in a, H_j^{-} \in a \\ b_j & \text{if } H_j^{+} \in a, H_j^{-} \notin a \\ d_j & \text{if } H_j^{+} \notin a, H_j^{-} \in a \end{cases} \tag{24}$$

We obtain $b_0 + c_0$ with the inclusion-exclusion formula applied to the minimal positive paths, thus $b_0 = I(A^{+}) - c_0$. Similarly, the degree of disbelief is $d_0 = I(A^{-}) - c_0$ and finally we obtain $i_0 = 1 - b_0 - d_0 - c_0$.

Example 3: Authenticity Verification with Inclusion-Exclusion Formula Fig. 5 shows an example scenario consisting of the first-hand statements $H_1, \dots H_6$ with associated confidence values. Entity E_A wants to know whether entity E_D is the holder of the key K_D and therefore requests the resulting confidence value for $H_0 = \text{Auth}(E_A, K_D, E_D)$.



j	H_j	b_j	i_j	d_j
1	$\text{Trust}(E_A, K_B, r_{\text{PKI}}, 0..1)$	0.95	0.05	0
2	$\text{Trust}(E_A, K_C, r_{\text{PKI}}, 0)$	0.85	0.15	0
3	$\text{Trust}(K_B, K_C, r_{\text{PKI}}, 0)$	0.9	0.1	0
4	$\text{Auth}(K_B, K_D, D_D)$	0.7	0.2	0.1
5	$\text{Auth}(K_C, K_D, D_D)$	0.8	0.2	0
6	$\text{Auth}(E_A, D_D, E_D)$	1	0	0

Fig. 5. Scenario of Example 3

Table 6. Propositions and applied inference rules in Example 3

Proposition	Inference rule	Origin
$H^+_{1,0} = \text{Trust}^+(E_A, K_B, r_{\text{PKI}}, 0, 1)$	-	from H_1
$H^+_{1,1} = \text{Trust}^+(E_A, K_B, r_{\text{PKI}}, 1, 1)$	-	from H_1
$H^+_{2,0} = \text{Trust}^+(E_A, K_C, r_{\text{PKI}}, 0, 1)$	-	from H_2
$H^+_{3,0} = \text{Trust}^+(K_B, K_C, r_{\text{PKI}}, 0, 1)$	-	from H_3
$H^+_4 = \text{Auth}^+(K_B, K_D, D_D)$	-	from H_4
$H^-_4 = \text{Auth}^-(K_B, K_D, D_D)$	-	from H_4
$H^+_5 = \text{Auth}^+(K_C, K_D, D_D)$	-	from H_5
$H^+_6 = \text{Auth}^+(E_A, D_D, E_D)$	-	from H_6
$H^+_7 = \text{Trust}^+(E_A, K_C, r_{\text{PKI}}, 0, 2)$	(7)	$H^+_{1,1}, H^+_{3,0} \vdash H^+_7$
$H^+_8 = \text{Auth}^+(E_A, K_D, D_D)$	(20)	$H^+_{1,0}, H^+_4 \vdash H^+_8$; $H^+_{2,0}, H^+_5 \vdash H^+_8$; $H^+_7, H^+_5 \vdash H^+_8$
$H^-_8 = \text{Auth}^-(E_A, K_D, D_D)$	(20)	$H^+_{1,0}, H^-_4 \vdash H^-_8$
$H^+_0 = \text{Auth}^+(E_A, K_D, E_D)$	(17)	$H^+_6, H^+_8 \vdash H^+_0$
$H^-_0 = \text{Auth}^-(E_A, K_D, E_D)$	(17)	$H^+_6, H^-_8 \vdash H^-_0$

First, we transform the trust statements from standard form into the internal form: H_1 is transformed into $H_{1,0} = \text{Trust}(E_A, K_B, r_{\text{PKI}}, 0, 1)$ and $H_{1,1} = \text{Trust}(E_A, K_B, r_{\text{PKI}}, 1, 1)$, H_2 into $H_{2,0} = \text{Trust}(E_A, K_C, r_{\text{PKI}}, 0, 1)$, etc. Then we create the set of propositions that represents a superposition of all possible worlds according to the confidence values of the statements (see Tab. 6, $H^+_{1,0}$, $\ldots H^+_6$). Next, we apply the inference rules to the proposition of this set (including the already derived propositions). The remaining rows of Tab. 6 list the derived propositions as well as the used inference rules and the premises. The transitive trust inference rule (7) allows for example to derive the new proposition $H^+_7 = \text{Trust}^+(E_A, K_C, r_{\text{PKI}}, 0, 2)$ from $H^+_{1,1}$ and $H^+_{3,0}$. Then the minimal positive and negative paths can be determined. We find the three positive paths $\{H^+_1, H^+_4, H^+_6\}$, $\{H^+_2, H^+_5, H^+_6\}$ and $\{H^+_1, H^+_3, H^+_5, H^+_6\}$ and one negative path $\{H^+_1, H^-_4, H^+_6\}$. We can thus construct the set of minimal conflicting paths: $\{H^+_1, H^+_2, H^-_4, H^+_5, H^+_6\}$, $\{H^+_1, H^+_3, H^-_4, H^+_5, H^+_6\}$ and $\{H^+_1, H^+_4, H^-_4, H^+_6\}$. The last path can be eliminated since $c_4 = 0$.

Next we compute the degrees of conflict, belief and disbelief with the inclusion-exclusion formula: $c_0 = b_1b_2d_4b_5b_6 + b_1b_3d_4b_5b_6 - b_1b_2b_3d_4b_5b_6 = 0.07486$, $b_0 = b_1b_4b_6 + b_2b_5b_6 + b_1b_3b_5b_6 - b_1b_2b_4b_5b_6 - b_1b_3b_4b_5b_6 - b_1b_2b_3b_5b_6 + b_1b_2b_3b_4b_5b_6 - c_0 =$

$0.92358 - c_0 = 0.84872$ and $d_0 = b_1 d_4 b_6 - c_0 = 0.095 - c_0 = 0.02014$. The degree of ignorance is $i_0 = 1 - b_0 - d_0 - c_0 = 0.05628$. Thus, the resulting confidence value for H_0 is $t_0 = (0.84872, 0.05628, 0.02014, 0.07486)$.

5.3 Comparison with Other Trust Models

The model of Maurer [4] does not allow to express degrees of disbelief. Therefore, conflict can never occur. In all scenarios in which Maurer's model can be applied it produces the same resulting confidence values as our model. Subjective Logic [5] can only be applied if the network is a directed series-parallel graph (e. g., Examples 1 and 2, but not Example 3). Credential Networks [11] can be applied only if at least one component of the confidence value (b_j, i_j or d_j) of each first-hand confidence value is zero. Subjective Logic and Credential Networks produce the same results as our model in all cases in which the models and their computation approaches can be applied and in which *no conflict* occurs (e. g., in Example 1). If conflicts are possible (e. g., in Examples 2 and 3), then the results generally differ from the results of our model because these models eliminate the probability mass associated with conflict.

Our model can thus be seen as an extension of Maurer's model, Subjective Logic and Credential Networks that overcomes the mentioned restrictions ($b > 0$, $i > 0$ and $d > 0$ is possible, no restriction to directed series-parallel graphs). However, we do not eliminate the degree of conflict, because this can cause counter-intuitive effects: Consider Example 2 (Sect. 5.2). If we choose $t_1 = t_2 = t_3 = t_4 = (1, 0, 0, 0)$ (full trust), then the resulting confidence value is $t_0 = (1, 0, 0, 0)$, too (in all models). If t_4 changes to $t_4 = (0.01, 0, 0.99, 0)$ (almost complete distrust), then the resulting confidence value in our model changes to $t_0 = (0.01, 0, 0, 0.99)$, which shows E_A that the trustworthiness of E_D is now highly disputed. However, in Subjective Logic and Credential Networks the resulting confidence value does not change. This gives E_A the wrong impression that there are no trustworthy entities who distrust E_D.

6 Computation Time

Computation time is a very (although not the most) important issue for reputation systems. The computation time to find the minimal paths appears to be uncritical because it is possible to check the inference rules efficiently and because the paths can be computed incrementally and in advance. Furthermore, the paths usually remain unchanged when the confidence values of existing opinions are updated.

The number of possible worlds to consider in the possible worlds algorithm increases exponentially with the number of *relevant first-hand statements*. It is therefore applicable if the number of relevant statements is small. It is important to emphasize that the computation time depends only on the number of *relevant* statements or paths, not on the *total* number. It is sufficient to consider only statements that are issued by the requester or by authentic entities or keys that

have been found to be trustworthy. Moreover, we can ignore all statements that are not part of a valid path, i. e., that do not contribute to answer the trust or authenticity request. Furthermore, most trust chains will not be longer than two or three statements. Therefore, the number of relevant statements or paths will usually be reasonably small. Although a trust and authenticity network similar to the PGP/GnuPG web of trust [1] can contain more than 100 000 trust and authenticity statements, the number of statements that are directly or indirectly (via valid paths) related to the requester will probably be below 100, and the number of statements that are part of valid paths from the requester to the requested statement is likely to be not higher than 10 or 20 in typical scenarios.

The number of possible worlds in the grouped possible worlds algorithm increases exponentially with the number of *groups*. Thus, the computation time can be reduced significantly if the statements can be grouped. Even large scenarios can be evaluated efficiently as long as the relevant statements can be subdivided into a small number of groups. In the inclusion-exclusion algorithm the number of summands increases exponentially with the number of relevant *paths*. This algorithm is therefore well suited for all scenarios with a small number of paths, even if the number of statements is large.

We illustrate the influence of the scenario (i. e., the number of relevant statements, paths and groups) on the computation time of the algorithms on two examples[5]. The scenarios are constructed in order to emphasize the large influence on the computation time and are not meant to be representative examples. For simplicity the scenarios consist only of trust statements between entities and refer to the same capability r. All confidence values contain degrees of belief, ignorance and disbelief ($b > 0, i > 0, d > 0$).

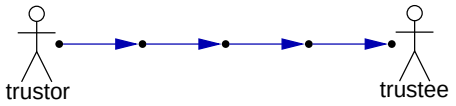


Fig. 6. Scenario of the simple chain example

The scenario with e entities in Fig. 6 consists of a simple chain and $e - 1$ trust statements with $h = 0..e$ recommendation hops. The possible worlds algorithm has to evaluate 3^{e-1} worlds. The scenario contains one positive and one negative path, therefore the inclusion-exclusion algorithm has to compute only two summands. The grouped possible world algorithm creates one group with three possible proposition-combinations: the positive path leads to belief, the negative path to disbelief, all other combinations lead to ignorance. It thus has to evaluate only three worlds. The diagram in Fig. 7 shows that the computation time of the possible world algorithm increases exponentially with the number of trust statements, whereas the computation time of the other algorithms increases linearly and thus remains insignificant even for long trust chains.

[5] implementation in Java 1.6; measurements on Intel Pentium M with 1.73 GHz

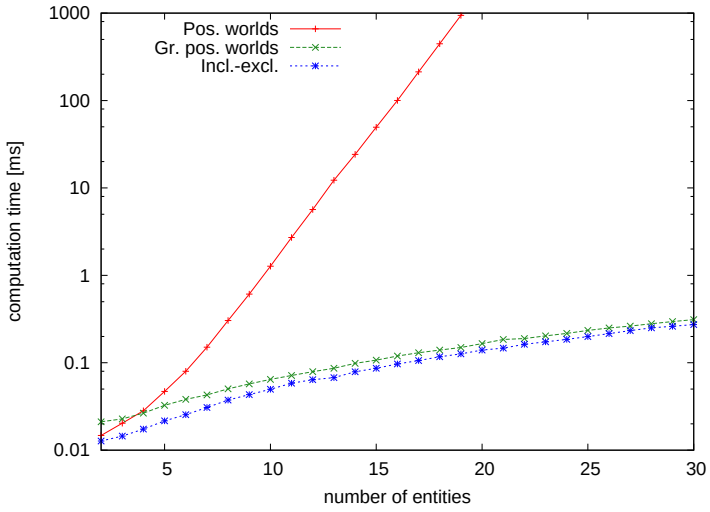


Fig. 7. Computation time in the simple chain example

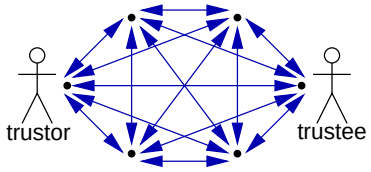


Fig. 8. Scenario of the full mesh example

The scenario in Fig. 8 is a full mesh. All entities trust each other for $h = 0..1$ hops. The number of relevant statements is $2e - 3$, the number of positive and negative paths is $e - 1$ and the number of conflicting paths is $(e - 1)(e - 2)$. Thus, the computation time of the possible worlds algorithm increases slower than of the inclusion-exclusion algorithm because the number of conflicting paths increases faster than the number of relevant statements (Fig. 9). The grouped possible worlds algorithm subdivides the statements into $e - 1$ groups, which reduces the number of possible worlds from 3^{2e-3} to 3^{e-1} worlds. Therefore the computation time remains acceptable for a larger number of entities than with the other algorithms.

The computation time heavily depends on the scenario. It is therefore difficult to give a general recommendation for one of the algorithms. It is possible that one algorithm outperforms an other by orders of magnitude in one scenario, but is much slower in an other scenario. The presented results and measurements in other scenarios suggest that the grouped possible worlds algorithm is in most scenarios the fastest (or at least close to the fastest) algorithm. However, further investigations are necessary.

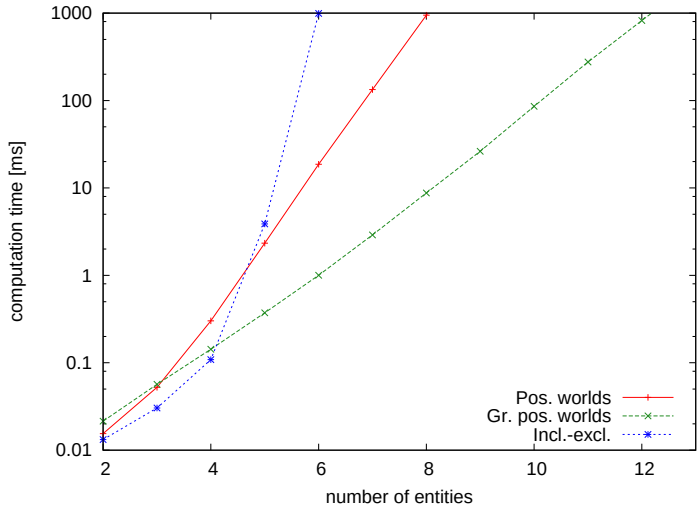


Fig. 9. Computation time in the full mesh example

An estimation for the computation time of the algorithms can be computed from the number of relevant statements, paths and groups. If the expected computation of all exact algorithms exceeds an acceptable limit, then a Monte-Carlo simulation can be used to compute an approximation. The number of iterations can be chosen according to the required accuracy and the acceptable computation time.

7 Summary and Conclusions

We presented a detailed model to represent trust and authenticity statements as well as confidence values and we proposed an integrated approach to reason with these statements and to compute resulting confidence values. The model distinguishes clearly between entities, descriptions and keys, allows multiple keys and descriptions per entity, distinguishes between functional and recommendation trust and allows to specify ranges of recommendation hops in trust statements. Confidence values allow to express degrees of belief, ignorance and disbelief. The system is able to reason with conflicting opinions because the presented inference rules are based on a paraconsistent logic. The computation of the resulting confidence values is based on a probability theoretical model in order to produce consistent results. In conflict-free scenarios our model is consistent with the Model of Maurer, Subjective Logic and Credential Networks, but overcomes several restrictions of these models. In conflicting scenarios we do not eliminate the degree of conflict in order to avoid counter-intuitive effects caused by re-normalizations.

We proposed different algorithms to implement the confidence value computation. Although the computation time increases exponentially with the number

of relevant statements, groups or paths it can be expected that an acceptable computation time can be reached in the majority of realistic scenarios. In the other cases, we propose to compute an approximation with Monte-Carlo simulations.

Acknowledgements This work was funded by the German Research Foundation (DFG) through the Collaborative Research Center (SFB) 627.

References

1. Ashley, J.M., Copeland, M., Grahn, J., Wheeler, D.A.: The GNU Privacy Handbook. The Free Software Foundation. (1999)
2. Grandison, T., Sloman, M.: A Survey of Trust in Internet Application. IEEE Communications Surveys & Tutorials **3**(4) (2000) 2–16
3. Marsh, S.P.: Formalising Trust as a Computational Concept. PhD thesis, Department of Mathematics and Computer Science, University of Stirling (1994)
4. Maurer, U.: Modelling a Public-Key Infrastructure. In: Proc. 1996 European Symposium on Research in Computer Security (ESORICS' 96). Volume 1146 of Lecture Notes in Computer Science., Springer-Verlag (1996) 325–350
5. Jøsang, A.: Artificial Reasoning with Subjective Logic. In: Proceedings of the Second Australian Workshop on Commonsense Reasoning. (1997)
6. Haenni, R., Kohlas, J., Lehmann, N. In: Probabilistic Argumentation Systems. Volume 5 (Algorithms for Uncertainty and Defeasible Reasoning) of Handbook of Defeasible Reasoning and Uncertainty Management Systems. Springer (2000) 221–288
7. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: Proceedings of the 12th International Conference on World Wide Web. (2003) 640–651
8. Demolombe, R.: Reasoning about trust: A formal logical framework. In: Proceedings of the Second International Conference of Trust Management (iTrust 2004). (2004) 291–303
9. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. In: Decision Support Systems. (2007)
10. Kohlas, R.: Decentralized Trust Evaluation and Public-Key Authentication. PhD thesis, Universität Bern (2007)
11. Jonczy, J., Haenni, R.: Credential Networks: a General Model for Distributed Trust and Authenticity Management. In: PST. (2005) 101–112
12. Jøsang, A., Gray, E., Kinateder, M.: Simplification and Analysis of Transitive Trust Networks. Web Intelligence and Agent Systems Journal (2006) 139–161
13. Zadeh, L.A.: Review of Books: A Mathematical Theory of Evidence. The AI Magazine **5**(3) (1984) 81–83
14. Shafer, G.: A Mathematical Theory of Evidence. Princeton Univ. Press (1976)
15. Gutscher, A.: A Trust Model for an Open, Decentralized Reputation System. In: Proceedings of the Joint iTrust and PST Conferences on Privacy Trust Management and Security (IFIPTM 2007). (2007) 285–300
16. Gutscher, A.: Reasoning with Uncertain and Conflicting Opinions in Open Reputation Systems. In: Proceedings of the Fourth International Workshop on Security and Trust Management (STM 2008). (2008)

Manual vs. Automated Vulnerability Assessment: A Case Study

James A. Kupsch and Barton P. Miller

Computer Sciences Department, University of Wisconsin, Madison, WI, USA
{kupsch,bart}@cs.wisc.edu*

Abstract. The dream of every software development team is to assess the security of their software using only a tool. In this paper, we attempt to evaluate and quantify the effectiveness of automated source code analysis tools by comparing such tools to the results of an in-depth manual evaluation of the same system. We present our manual vulnerability assessment methodology, and the results of applying this to a major piece of software. We then analyze the same software using two commercial products, Coverity Prevent and Fortify SCA, that perform static source code analysis. These tools found only a few of the fifteen serious vulnerabilities discovered in the manual assessment, with none of the problems found by these tools requiring a deep understanding of the code. Each tool reported thousands of defects that required human inspection, with only a small number being security related. And, of this small number of security-related defects, there did not appear to be any that indicated significant vulnerabilities beyond those found by the manual assessment.

1 Introduction

While careful design practices are necessary to the construction of secure systems, they are only part of the process of designing, building, and deploying such a system. To have high confidence in a system's security, a systematic assessment of its security is needed before deploying it. Such an assessment, performed by an entity independent of the development team, is a crucial part of development of any secure system. Just as no serious software project would consider skipping the step of having their software evaluated for correctness by an independent testing group, a serious approach to security requires independent assessment for vulnerabilities. At the present time, such an assessment is necessarily an expensive task as it involves a significant commitment of time from a security analyst. While using automated tools is an attractive approach to making this task less labor intensive, even the best of these tools appear limited in the kinds of vulnerabilities that they can identify.

* D. Chadwick, I. You and H. Chang (Eds.): Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST2009), Purdue University, West Lafayette, USA, June 16, 2009.

In this paper, we attempt to evaluate and quantify the effectiveness of automated source code vulnerability assessment tools [1] by comparing such tools to the results of an in-depth manual evaluation of the same system.

We started with a detailed vulnerability assessment of a large, complex, and widely deployed distributed system called Condor [11, 15, 2]. Condor is a system that allows the scheduling of complex tasks over local and widely distributed networks of computers that span multiple organizations. It handles scheduling, authentication, data staging, failure detection and recovery, and performance monitoring. The assessment methodology that we developed, called *First Principles Vulnerability Assessment* (FPVA), uses a top-down resource centric approach to assessment that attempts to identify the components of a systems that are most at risk, and then identifying vulnerabilities that might be associated with them. The result of such an approach is to focus on the places in the code where high value assets might be attacked (such as critical configuration files, parts of the code that run at high privilege, or security resources such as digital certificates). This approach shares many characteristics with techniques such as Microsoft's threat modeling [14] but with a key difference: we start from high valued assets and work outward to derive vulnerabilities rather than start with vulnerabilities and then see if they lead to a serious exploit.

In 2005 and 2006, we performed an analysis on Condor using FPVA, resulting in the discovery of fifteen major vulnerabilities. These vulnerabilities were all confirmed by developing sample exploit code that could trigger each one.

More recently, we made an informal survey of security practitioners in industry, government, and academia to identify what were the best automated tools for vulnerability assessment. Uniformly, the respondents identified two highly-regarded commercial tools: Coverity Prevent [5] and Fortify Source Code Analyzer (SCA) [8] (while these companies have multiple products, in the remainder of this paper we will refer to Coverity Prevent and Fortify Source Code Analyzer as "Coverity" and "Fortify" respectively). We applied these tools to the same version of Condor as was used in the FPVA study to compare the ability of these tools to find serious vulnerabilities (having a low false negative rate), while not reporting a significant number of false vulnerabilities or vulnerabilities with limited exploit value (having a low false positive rate).

The most significant findings from our comparative study were:

1. Of the 15 serious vulnerabilities found in our FPVA study of Condor, Fortify found six and Coverity only one.
2. Both Fortify and Coverity had significant false positive rates with Coverity having a lower false positive rate. The volume of these false positives were significant enough to have a serious impact on the effectiveness of the analyst.
3. In the Fortify and Coverity results, we found no significant vulnerabilities beyond those identified by our FPVA study. (This was not an exhaustive study, but did thoroughly cover the problems that the tools identified as most serious.)

To be fair, we did not expect the automated tools to find all the problems that could be found by an experienced analyst using a systematic methodology.

The goals of this study were (1) to try to identify the places where an automated analysis can simplify the assessment task, and (2) start to characterize the kind of problems *not* found by these tools so that we can develop more effective automated analysis techniques.

One could claim that the results of this study are not surprising, but there are no studies to provide strong evidence of the strengths and weaknesses of software assessment tools. The contributions of this paper include:

1. showing clearly the limitations of current tools,
2. presenting manual vulnerability assessment as a required part of a comprehensive security audit, and
3. creating a reference set of vulnerabilities to perform apples-to-apples comparisons.

In the next section, we briefly describe our FPVA manual vulnerability assessment methodology, and then in Section 3, we describe the vulnerabilities that were found when we applied FPVA to the Condor system. Next, in Section 4, we describe the test environment in which the automated tools were run and how we applied Coverity and Fortify to Condor. Section 5 describes the results from this study along with a comparison of these results to our FPVA analysis. The paper concludes with comments on how the tools performed in this analysis.

2 First Principles Vulnerability Assessment (FPVA)

This section briefly describes the methodology used to find most of the vulnerabilities used in this study. Most of the vulnerabilities in Condor were discovered using a manual vulnerability assessment we developed at the University of Wisconsin called first principles vulnerability assessment (FPVA). The assessment was done independently, but in cooperation with the Condor development team.

FPVA consists of four analyses where each relies upon the prior steps to focus the work in the current step. The first three steps, architectural, resource, and trust and privilege analyses are designed to assist the assessor in understand the operation of the system under study. The final step, the component evaluation, is where the search for vulnerabilities occurs using the prior analyses and code inspection. This search focuses on likely high-value resources and pathways through the system.

The architectural analysis is the first step of the methodology and is used to identify the major structural components of the system, including hosts, processes, external dependencies, threads, and major subsystems. For each of these components, we then identify their high-level function and the way in which they interact, both with each other and with users. Interactions are particularly important as they can provide a basis to understand the information flow through, and how trust is delegated through the system. The artifact produced at this stage is a document that diagrams the structure of the system and the interactions.

The next step is the resource analysis. This step identifies the key resources accessed by each component, and the operations supported on these resources. Resources include things such as hosts, files, databases, logs, CPU cycles, storage, and devices. Resources are the targets of exploits. For each resource, we describe its value as an end target (such as a database with personnel or proprietary information) or as an intermediate target (such as a file that stores access-permissions). The artifact produced at this stage is an annotation of the architectural diagrams with resource descriptions.

The third step is the trust and privilege analysis. This step identifies the trust assumptions about each component, answering such questions as how are they protected and who can access them? For example, a code component running on a client's computer is completely open to modification, while a component running in a locked computer room has a higher degree of trust. Trust evaluation is also based on the hardware and software security surrounding the component. Associated with trust is describing the privilege level at which each executable component runs. The privilege levels control the extent of access for each component and, in the case of exploitation, the extent of damage that it can directly accomplish. A complex but crucial part of trust and privilege analysis is evaluating trust delegation. By combining the information from steps 1 and 2, we determine what operations a component will execute on behalf of another component. The artifact produced at this stage is a further labeling of the basic diagrams with trust levels and labeling of interactions with delegation information.

The fourth step is component evaluations, where components are examined in depth. For large systems, a line-by-line manual examination of the code is infeasible, even for a well-funded effort. The step is guided by information obtained in steps 1–3, helping to prioritize the work so that high-value targets are evaluated first. Those components that are part of the communication chain from where user input enters the system to the components that can directly control a strategic resource are the components that are prioritized for assessment. There are two main classifications of vulnerabilities: design (or architectural) flaws, and implementation bugs [12]. Design flaws are problems with the architecture of the system and often involve issues of trust, privilege, and data validation. The artifacts from steps 1–3 can reveal these types of problems or greatly narrow the search. Implementation bugs are localized coding errors that can be exploitable. Searching the critical components for these types of errors results in bugs that have a higher probability of exploit as they are more likely to be in the chain of processing from users input to critical resource. Also the artifacts aid in determining if user input can flow through the implementation bug to a critical resource and allow the resource to be exploited.

3 Results of the Manual Assessment

Fifteen vulnerabilities in the Condor project had been discovered and documented in 2005 and 2006. Most of these were discovered through a systematic,

manual vulnerability assessment using the FPVA methodology, with a couple of these vulnerabilities being reported by third parties. Table 1 lists each vulnerability along with a brief description. A complete vulnerability report that includes full details of each vulnerability is available from the Condor project [4] for most of the vulnerabilities.

The types of problems discovered included a mix of implementation bugs and design flaws. The following vulnerabilities are caused by implementation bugs: CONDOR-2005-0003 and CONDOR-2006-000{1,2,3,4,8,9}. The remaining vulnerabilities are caused by design flaws. The vulnerability CONDOR-2006-0008 is unusual in that it only exists on certain older platforms that only provide an unsafe API to create a temporary file.

Table 1: Summary of Condor vulnerabilities discovered in 2005 and 2006 and whether Fortify or Coverity discovered the vulnerability.

Vuln. Id	Fortify	Coverity	Vulnerability Description	Tool Discoverable?
CONDOR-2005-0001	no	no	A path is formed by concatenating three pieces of user supplied data with a base directory path to form a path to to create, retrieve or remove a file. This data is used as is from the client which allows a directory traversal [7] to manipulate arbitrary file locations.	Difficult. Would have to know path was formed from untrusted data, not validated properly, and that a directory traversal could occur. Could warn about untrusted data used in a path.
CONDOR-2005-0002	no	no	This vulnerability is a lack of authentication and authorization. This allows impersonators to manipulate checkpoint files owned by others.	Difficult. Would have to know that there should be an authentication and authorization mechanism, which is missing.
CONDOR-2005-0003	yes	no	This vulnerability is a command injection [7] resulting from user supplied data used to form a string. This string is then interpreted by `/bin/sh` using a `fork` and `execl("/bin/sh", "-c", command)`.	Easy. Should consider network and file data as tainted and all the parameters to execl as sensitive.
CONDOR-2005-0004	no	no	This vulnerability is caused by the insecure owner of a file used to store persistent overridden configuration entries. These configuration entries can cause arbitrary executable files to be started as root.	Difficult. Would have to track how these configuration setting flow into complex data structure before use, both from files that have the correct ownership and permissions and potentially from some that do not.
CONDOR-2005-0005	no	no	This vulnerability is caused by the lack of an integrity [7] check on checkpoints (a representation of a running process that can be restarted) that are stored on a checkpoint server. Without a way of ensuring the integrity of the checkpoint, the checkpoint file could be tampered with to run malicious code.	Difficult. This is a high level design flaw that a particular server should not be trusted.

Table 1 – Continued.

Vuln. Id	Fortify	Coverity	Vulnerability Description	Tool Discoverable?
CONDOR-2005-0006	no	no	Internally the Condor system will not run user's jobs with the user id of the root account. There are other accounts on machines which should also be restricted, but there are no mechanisms to support this.	Difficult. Tool would have to know which accounts should be allowed to be used for what purposes.
CONDOR-2006-0001	yes	no	The stork subcomponent of Condor, takes a URI for a source and destination to move a file. If the destination file is local and the directory does not exist the code uses the `system` function to create it without properly quoting the path. This allows a command injection to execute arbitrary commands. There are 3 instances of this vulnerability.	Easy. The string used as the parameter to `system` comes fairly directly from an untrusted argv value.
CONDOR-2006-0002	yes	no	The stork subcomponent of Condor, takes a URI for a source and destination to move a file. Certain combinations of schemes of the source and destination URIs cause stork to call helper applications using a string created with the URIs, and without properly quoting them. This string is then passed to `popen`, which allows a command injection to execute arbitrary commands. There are 6 instances of this vulnerability.	Easy. The string used as the parameter to `popen` comes from a substring of an untrusted `argv` value.
CONDOR-2006-0003	yes	no	Condor class ads allow functions. A function that can be enabled, executes an external program whose name and arguments are specified by the user. The output of the program becomes the result of the function. The implementation of the function uses `popen` without properly quoting the user supplied data.	Easy. A call to `popen` uses data from an untrusted source such as the network or a file.
CONDOR-2006-0004	yes	no	Condor class ads allow functions. A function that can be enabled, executes an external program whose name and arguments are specified by the user. The path of the program to run is created by concatenating the script directory path with the name of the script. Nothing in the code checks that the script name cannot contain characters that allows for a directory traversal.	Easy. A call to `popen` uses data from an untrusted source such as the network or a file. It would be difficult for a tool to determine if an actual path traversal is possible.
CONDOR-2006-0005	no	no	This vulnerability involves user supplied data being written as records to a file with the file later reread and parsed into records. Records are delimited by a new line, but the code does not escape new lines or prevent them in the user supplied data. This allows additional records to be injected into the file.	Difficult. Would have to deduce the format of the file and that the injection was not prevented.

Table 1 – Continued.

Vuln. Id	Fortify	Coverity	Vulnerability Description	Tool Discoverable?
CONDOR-2006-0006	no	no	This vulnerability involves an authentication mechanism that assumes a file with a particular name and owner can be created only by the owner or the root user. This is not true as any user can create a hard link, in a directory they write, to any file and the file will have the permissions and owner of the linked file, invalidating this assumption.[10]	Difficult. Would require the tool to understand why the existence and properties are being checked and that they can be attacked in certain circumstances.
CONDOR-2006-0007	no	no	This vulnerability is due to a vulnerability in OpenSSL [6] and requires a newer version of the library to mitigate.	Difficult. The tool would have to have a list of vulnerable library versions. It would also be difficult to discover if the tool were run on the library code as the defect is algorithmic.
CONDOR-2006-0008	no	no	This vulnerability is caused by using a combination of the functions `tmpnam` and `open` to try and create a new file. This allows an attacker to use a classic time of check, time of use (TOCTOU) [7] attack against the program to trick the program into opening an existing file. On platforms that have the function `mkstemp`, it is safely used instead.	Hard. The unsafe function is only used (compiled) on a small number of platforms. This would be easy for a tool to detect if the unsafe version is compiled. Since the safe function `mkstemp` existed on the system, the unsafe version was not seen by the tools.
CONDOR-2006-0009	yes	yes	This vulnerability is caused by user supplied values being placed in a fixed sized buffer that lack bounds checks. The user can then cause a buffer overflow [16] that can result in a crash or stack smashing attack.	Easy. No bounds check is performed when writing to a fixed sized buffer (using the dangerous function `strcpy`) and the data comes from an untrusted source.
Total	6	1	out of 15 total vulnerabilities	

4 Setup and Running of Study

4.1 Experiment Setup

To perform the evaluation of the Fortify and Coverity tools, we used the same version of Condor, run in the same environment, as was used in our FPVA analysis. The version of the source code, platform and tools used in this test were as follows:

1. Condor 6.7.12
   (a) with 13 small patches to allow compilation with newer GNU compiler collection (gcc) [9];

   (b) built as a clipped [3] version, i.e., no standard universe, Kerberos, or Quill as these would not build without extensive work on the new platform and tool chain.
2. gcc (GCC) 3.4.6 20060404 (Red Hat 3.4.6-10)
3. Scientific Linux SL release 4.7 (Beryllium) [13]
4. Fortify SCA 5.1.0016 rule pack 2008.3.0.0007
5. Coverity Prevent 4.1.0

To get both tools to work required using a version of gcc that was newer than had been tested with Condor 6.7.12. This necessitated 13 minor patches to prevent gcc from stopping with an error. Also this new environment prevented building Condor with standard universe support, Kerberos, and Quill. None of these changes affected the presence of the discovered vulnerabilities.

The tools were run using their default settings except Coverity was passed the flag `--all` to enable all the analysis checkers (Fortify enables all by default).

4.2 Tool Operation

Both tools operate in a similar three step fashion: gather build information, analyze, and present results. The build information consists of the files to compile, and how they are used to create libraries and executable files. Both tools make this easy to perform by providing a program that takes as arguments the normal command used to build the project. The information gathering tool monitors the build's commands to create object files, libraries and executables.

The second step performs the analysis. This step is also easily completed by running a program that takes the result of the prior step as an input. The types of checkers to run can also be specified. The general term *defect* will be used to describe the types of problems found by the tools as not all problems result in a vulnerability.

Finally, each tool provides a way to view the results. Coverity provides a web interface, while Fortify provides a stand-alone application. Both viewers allow the triage and management of the discovered defects. The user can change attributes of the defect (status, severity, assigned developer, etc.) and attach additional information. The status of previously discovered defects in earlier analysis runs is remembered, so the information does not need to be repeatedly entered.

Each tool has a collection of checkers that categorize the type of defects. The collection of checkers depends on the source language and the options used during the analysis run. Fortify additionally assigns each defect a severity level of Critical, Hot, Warning and Info. Coverity does not assign a severity, but allows one to be assigned by hand.

4.3 Tool Output Analysis

After both tools were run on the Condor source, the results from each tool were reviewed against the known vulnerabilities and were also sampled to look for vulnerabilities that were not found using the FPVA methodology.

The discovered vulnerabilities were all caused by code at one or at most a couple of a lines or functions. Both tools provided interfaces that allowed browsing the found defects by file and line. If the tool reported a defect at the same location in the code and of the correct type the tool was determined to have found the vulnerability.

The defects discovered by the tools were also sampled to determine if the tools discovered other vulnerabilities and to understand the qualities of the defects. The sampling was weighted to look more at defects found in higher impact locations in the code and in the categories of defects that are more likely to impact security. We were unable to conduct an exhaustive review the results due to time constraints and the large number of defects presented by the tools.

5 Results of the Automated Assessment

This section describes the analysis of the defects found by Coverity and Fortify. We first compare the results of the tools to the vulnerabilities found by FPVA. Next we empirically look at the false positive and false negative rates of the tools and the reasons behind these. Finally we offer some commentary on how the tools could be improved.

Fortify discovered all the vulnerabilities we expected it to find, those caused by implementation bugs, while Coverity only found a small subset. Each tool reported a large number of defects. Many of these are indications of potential correctness problems, but out of those inspected none appeared to be a significant vulnerability.

5.1 Tools Compared to FPVA Results

Table 1 presents each vulnerability along with an indication if Coverity or Fortify also discovered the vulnerability.

Out of the fifteen known vulnerabilities in the code, Fortify found six of them, while Coverity only discovered one of them. Vulnerability CONDOR-2006-0001 results from three nearly identical vulnerability instances in the code, and vulnerability CONDOR-2006-0002 results from six nearly identical instances. Fortify discovered all instances of these two vulnerabilities, while Coverity found none of them.

All the vulnerabilities discovered by both tools were due to Condor's use of functions that commonly result in security problems such as `execl`, `popen`, `system` and `strcpy`. Some of the defects were traced to untrusted inputs being used in these functions. The others were flagged solely due to the dangerous nature of these functions. These vulnerabilities were simple implementation bugs that could have been found by using simple scripts based on tools such as `grep` to search for the use of these functions.

5.2 Tool Discovered Defects

Table 2 reports the defects that we found when using Fortify, dividing the defects into categories with a count of how often each defect category occurred. Table 3 reports the defects found when using Coverity. The types of checkers that each tool reports are not directly comparable, so no effort was done to do so. Fortify found a total of 15,466 defects while Coverity found a total of 2,686. The difference in these numbers can be attributed to several reasons:

1. differences in the analysis engine in each product;
2. Coverity creates one defect for each sink (place in the code where bad data is used in a way to cause the defect, and displays one example source to sink path), while Fortify has one defect for each source/sink pair; and
3. Coverity seems to focus on reducing the number of false positives at the risk of missing true positives, while Fortify is more aggressive in reporting potential problems resulting in more false positives.

From a security point of view, the sampled defects can be categorized in order of decreasing importance as follows:

1. *Security Issues.* These problems are exploitable. Other than the vulnerabilities also discovered in the FPVA (using tainted data in risk functions), the only security problems discovered were of a less severe nature. They included denial of service issues due to the dereference of null pointers, and resource leaks.
2. *Correctness Issues.* These defects are those where the code will malfunction, but the security of the application is not affected. These are caused by problems such as (1) a buffer overflow of a small number of bytes that may cause incorrect behavior, but do not allow execution of arbitrary code or other security problems, (2) the use of uninitialized variables, or (3) the failure to check the status of certain functions.
3. *Code Quality Issues.* Not all the defects found are directly security related, such as Coverity's parse warnings (those starting with PW), dead code and unused variables, but they are a sign of code quality and can result in security problem in the right circumstances.

Due to the general fragility of code, small changes in code can easily move a defect from one category to another, so correcting the non-security defects could prevent future vulnerabilities.

5.3 False Positives

False positives are the defects that the tool reports, but are not actually defects. Many of these reported defects are items that should be repaired as they often are caused by poor programming practices that can easily develop into a true defect during modifications to the code. Given the finite resources in any assessment activity, these types of defects are rarely fixed. Ideally, a tool such

Table 2. Defect counts reported by Fortify by type and severity level.

Vuln Type	Total	Critical	Hot	Warning	Info
Buffer Overflow	2903	0	1151	391	1361
Buffer Overflow: Format String	1460	0	995	465	0
Buffer Overflow: Format String (%f/%F)	75	0	42	33	0
Buffer Overflow: Off-by-One	4	0	4	0	0
Command Injection	108	0	81	15	12
Dangerous Function	3	3	0	0	0
Dead Code	589	0	0	0	589
Denial of Service	2	0	0	2	0
Double Free	33	0	0	33	0
Format String	105	0	27	24	54
Format String: Argument Type Mismatch	3	0	0	3	0
Heap Inspection	16	0	0	0	16
Illegal Pointer Value	1	0	0	0	1
Insecure Randomness	5	0	0	0	5
Insecure Temporary File	6	0	0	1	5
Integer Overflow	1168	0	0	274	894
Memory Leak	906	0	0	906	0
Memory Leak: Reallocation	6	0	0	6	0
Missing Check against Null	670	0	0	670	0
Null Dereference	263	0	0	263	0
Obsolete	78	0	0	0	78
Often Misused: Authentication	24	0	0	0	24
Often Misused: File System	5	0	0	0	5
Often Misused: Privilege Management	15	0	0	0	15
Out-of-Bounds Read	2	0	0	2	0
Out-of-Bounds Read: Off-by-One	3	0	0	3	0
Path Manipulation	463	0	0	444	19
Poor Style: Redundant Initialization	14	0	0	0	14
Poor Style: Value Never Read	120	0	0	0	120
Poor Style: Variable Never Used	277	0	0	0	277
Process Control	1	0	1	0	0
Race Condition: File System Access	92	0	0	92	0
Race Condition: Signal Handling	15	0	0	15	0
Redundant Null Check	108	0	0	108	0
Resource Injection	58	0	0	58	0
Setting Manipulation	28	0	0	28	0
String Termination Error	4708	0	0	3702	1006
System Information Leak	760	0	0	458	302
Type Mismatch: Signed to Unsigned	2	0	0	0	2
Unchecked Return Value	137	0	0	0	137
Uninitialized Variable	125	0	0	0	125
Unreleased Resource	82	0	0	82	0
Unreleased Resource: Synchronization	2	0	0	2	0
Use After Free	21	0	0	21	0
Total	15466	3	2301	8101	5061

Table 3. Defect counts reported by Coverity by type.

Total	Vulnerability Type	Total	Vulnerability Type
2	ARRAY_VS_SINGLETON	38	REVERSE_INULL
1	ATOMICITY	0	REVERSE_NEGATIVE
0	BAD_ALLOC_ARITHMETIC	842	SECURE_CODING
0	BAD_ALLOC_STRLEN	4	SECURE_TEMP
0	BAD_COMPARE	2	SIZECHECK
0	BAD_FREE	0	SLEEP
1	BAD_OVERRIDE	378	STACK_USE
1	BUFFER_SIZE	1	STREAM_FORMAT_STATE
32	BUFFER_SIZE_WARNING	2	STRING_NULL
5	CHAR_IO	147	STRING_OVERFLOW
82	CHECKED_RETURN	10	STRING_SIZE
0	CHROOT	6	TAINTED_SCALAR
2	CTOR_DTOR_LEAK	43	TAINTED_STRING
29	DEADCODE	26	TOCTOU
5	DELETE_ARRAY	0	UNCAUGHT_EXCEPT
0	DELETE_VOID	330	UNINIT
0	EVALUATION_ORDER	96	UNINIT_CTOR
40	FORWARD_NULL	9	UNREACHABLE
2	INFINITE_LOOP	31	UNUSED_VALUE
0	INTEGER_OVERFLOW	12	USE_AFTER_FREE
0	INVALIDATE_ITERATOR	5	VARARGS
0	LOCK	0	WRAPPER_ESCAPE
0	LOCK_FINDER	1	PW.BAD_MACRO_REDEF
3	MISSING_LOCK	5	PW.BAD_PRINTF_FORMAT_STRING
17	MISSING_RETURN	56	PW.IMPLICIT_FUNC_DECL
17	NEGATIVE_RETURNS	1	PW.IMPLICIT_INT_ON_MAIN
18	NO_EFFECT	18	PW.INCLUDE_RECURSION
32	NULL_RETURNS	20	PW.MISSING_TYPE_SPECIFIER
4	OPEN_ARGS	46	PW.NON_CONST_PRINTF_FORMAT_STRING
4	ORDER_REVERSAL	2	PW.PARAMETER_HIDDEN
3	OVERRUN_DYNAMIC	20	PW.PRINTF_ARG_MISMATCH
30	OVERRUN_STATIC	10	PW.QUALIFIER_IN_MEMBER_DECLARATION
3	PASS_BY_VALUE	2	PW.TOO_FEW_PRINTF_ARGS
1	READLINK	7	PW.TOO_MANY_PRINTF_ARGS
150	RESOURCE_LEAK	11	PW.UNRECOGNIZED_CHAR_ESCAPE
0	RETURN_LOCAL	21	PW.USELESS_TYPE_QUALIFIER_ON-_RETURN_TYPE
		2686	Total

as Fortify or Coverity is run regularly during the development cycle, allowing the programmers to fix such defects as they appear (resulting in a lower false positive rate). In reality, these tools are usually applied late in the lifetime of a software system.

Some of the main causes of false positives found in this study are the following:

1. Non-existent code paths due to functions that never return due to an `exit` or `exec` type function. Once in a certain branch, the program is guaranteed to never execute any more code in the program due to these functions and the way that code is structured, but the tool incorrectly infers that it can continue past this location.
2. Correlated variables, where the value of one variable restricts the set of values the other can take. This occurs when a function returns two values, or two fields of a structure. For instance, a function could return two values, one a pointer and the other a boolean indicating that the pointer is valid; if the boolean is checked before the dereferencing of the pointer, the code is correct, but if the tool does not track the correlation it appears that a null pointer dereference could occur.
3. The value of a variable is restricted to a subset of the possible values, but is not deduced by the tool. For instance, if a function can return only two possible errors, and a switch statement only handles these exact two errors, the code is correct, but a defect is produced due to not all possible errors being handled.
4. Conditions outside of the function prevent a vulnerability. This is caused when the tool does not deduce that:
   (a) Data read from certain files or network connections should be trusted due to file permissions or prior authentication.
   (b) The environment is secure due to a trusted parent process securely setting the environment.
   (c) A variable is constrained to safe values, but it is hard to deduce.

The false positives tend to cluster in certain checkers (and severity levels in Fortify). Some checkers will naturally have less reliability than others. The other cause of the cluster is due to developers repeating the same idiom throughout the code. For instance, almost all of the 330 UNINIT defects that Coverity reports are false positives due to a recurring idiom.

Many of these false positive defects are time bombs waiting for a future developer to unwittingly make a change somewhere in the code that affects the code base to now allow the defect to be true. A common example of this is a string buffer overflow, where the values placed in the buffer are currently too small in aggregate to overflow the buffer, but if one of these values is made bigger or unlimited in the future, the program now has a real defect.

Many of the false positives can be prevented by switching to a safer programming idiom, where it should take less time to make this change than for a developer to determine if the defect is actually true or false. The uses of `sprintf`, `strcat` and `strcpy` are prime examples of this.

5.4 False Negatives

False negatives are defects in the code that the tool did not report. These defects include the following:

1. Defects that are high level design flaws. These are the most difficult defects for a tool to detect as the tool would have to understand design requirements not present in the code.
2. The dangerous code is not compiled on this platform. The tools only analyze the source code seen when the build information gathering step is run. The tools ignore files that were not compiled and parts of files that were conditionally excluded. A human inspecting the code can easily spot problems that occur in different build configurations.
3. Tainted data becomes untainted. The five vulnerabilities that Fortify found, but Coverity did not were caused by Coverity only reporting an issue with functions such as `execl`, `popen` and `system` if the data is marked as tainted. The tainted property of strings is only transitive when calling certain functions such as `strcpy` or `strcat`. For instance, if a substring is copied byte by byte, Coverity does not consider the destination string as tainted.
4. Data flows through a pointer to a heap data structure, that the tool cannot track.

Some of these are defects that a tool will never find, while some of these will hopefully be found by tools in the future as the quality of their analysis improves.

5.5 Improving the Tool's Results

Both tools allow the analyst to provide more information to the tool to increase the tools accuracy. This information is described by placing annotations in the source code, or a simple description of the additional properties can be imported into the tools analysis model.

A simple addition could be made to Coverity's model to flag all uses of certain system calls as unsafe. This would report all the discovered vulnerabilities that Fortify found along with all the false positives for these types of defects.

6 Conclusion

This study demonstrates the need for manual vulnerability assessment performed by a skilled human as the tools did not have a deep enough understanding of the system to discover all of the known vulnerabilities.

There were nine vulnerabilities that neither tools discovered. In our analysis of these vulnerabilities, we did not expect a tool to find them due as they are caused by design flaws or were not present in the compiled code.

Out of the remaining six vulnerabilities, Fortify did find them all, and Coverity found a subset and should be able to find the others by adding a small model.

We expected a tool and even a simple to tool to be able to discover these vulnerabilities as they were simple implementation bugs.

The tools are not perfect, but they do provide value over a human for certain implementation bugs or defects such as resource leaks. They still require a skilled operator to determine the correctness of the results, how to fix the problem and how to make the tool work better.

7 Acknowledgments

This research funded in part by National Science Foundation grants OCI-0844219, CNS-0627501, and CNS-0716460.

References

[1] Brian Chess and Jacob West. *Secure Programming with Static Analysis*. Addison-Wesley, 2007.
[2] Condor Project. `http://www.cs.wisc.edu/condor`.
[3] Condor Team, University of Wisconsin. *Condor Manual*. `http://www.cs.wisc.edu/condor/manual`.
[4] Condor Vulnerability Reports. `http://www.cs.wisc.edu/condor/security/vulnerabilities`.
[5] Coverity Inc., Prevent. `http://www.coverity.com`.
[6] CVE-2006-4339. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-4339`, 2006. OpenSSL vulnerability.
[7] Mark Dowd, John McDonald, and Justin Schuh. *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*. Addison-Wesley, 2007.
[8] Fortify Software Inc., Source Code Analyzer (SCA). `http://www.fortify.com`.
[9] GNU Compiler Collection (gcc). `http://gcc.gnu.org`.
[10] James A. Kupsch and Barton P. Miller. How to Open a File and Not Get Hacked. In *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, pages 1196–1203, 2008.
[11] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - A Hunter of Idle Workstations. *Proc. 8th Intl Conf. on Distributed Computing Systems*, pages 104–111, June 1988.
[12] Gary McGraw. *Software Security*. Addison-Wesley, 2006.
[13] Scientific Linux, CERN and Fermi National Accelerator Laboratory. `http://www.scientificlinux.org`.
[14] Frank Swiderski and Window Snyder. *Threat Modeling*. Microsoft Press, 2004.
[15] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.
[16] John Viega and Gary McGraw. *Building Secure Software*. Addison-Wesley, 2002.