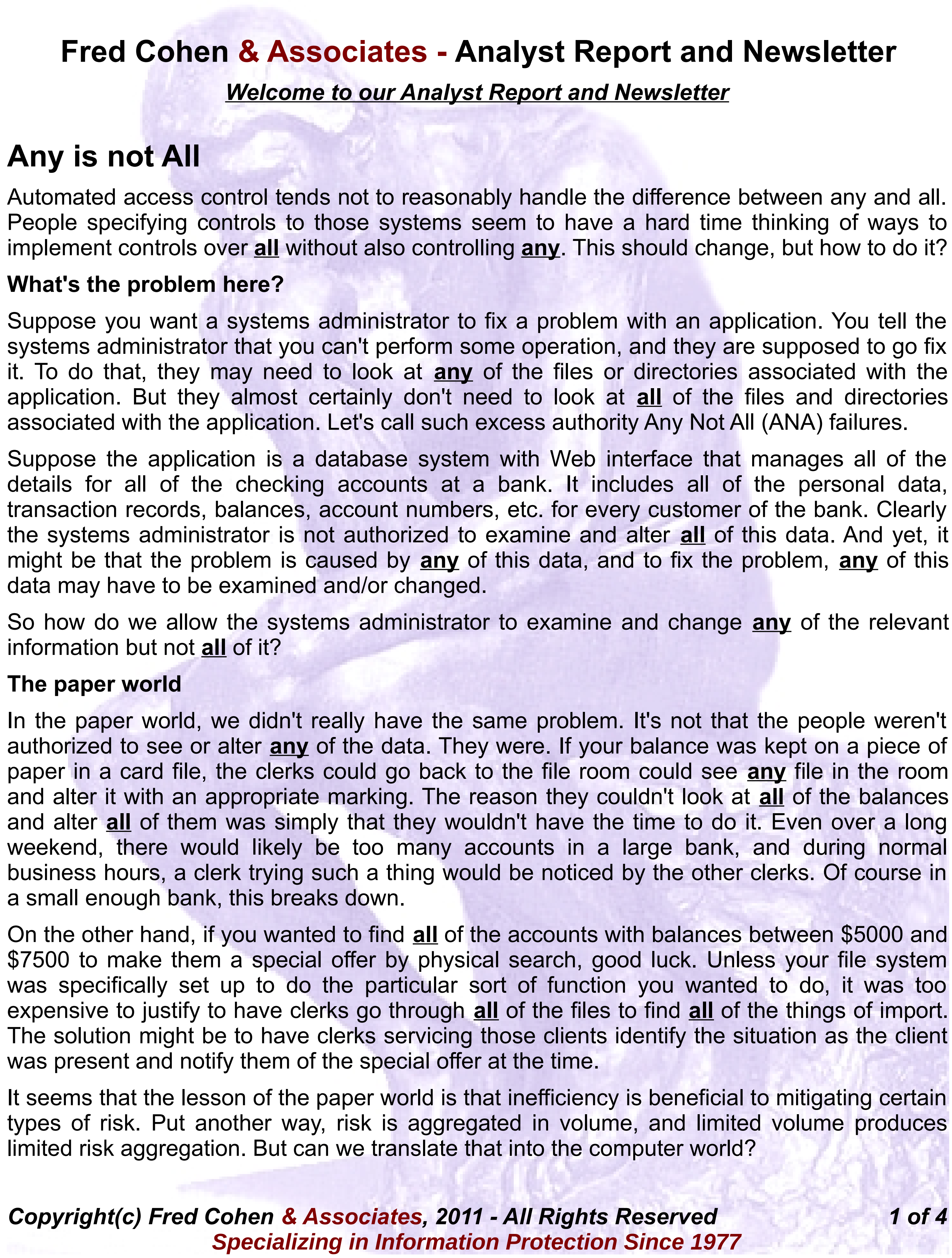

Fred Cohen & Associates - Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

Any is not All

Automated access control tends not to reasonably handle the difference between any and all. People specifying controls to those systems seem to have a hard time thinking of ways to implement controls over **all** without also controlling **any**. This should change, but how to do it?

What's the problem here?

Suppose you want a systems administrator to fix a problem with an application. You tell the systems administrator that you can't perform some operation, and they are supposed to go fix it. To do that, they may need to look at **any** of the files or directories associated with the application. But they almost certainly don't need to look at **all** of the files and directories associated with the application. Let's call such excess authority Any Not All (ANA) failures.

Suppose the application is a database system with Web interface that manages all of the details for all of the checking accounts at a bank. It includes all of the personal data, transaction records, balances, account numbers, etc. for every customer of the bank. Clearly the systems administrator is not authorized to examine and alter **all** of this data. And yet, it might be that the problem is caused by **any** of this data, and to fix the problem, **any** of this data may have to be examined and/or changed.

So how do we allow the systems administrator to examine and change **any** of the relevant information but not **all** of it?

The paper world

In the paper world, we didn't really have the same problem. It's not that the people weren't authorized to see or alter **any** of the data. They were. If your balance was kept on a piece of paper in a card file, the clerks could go back to the file room could see **any** file in the room and alter it with an appropriate marking. The reason they couldn't look at **all** of the balances and alter **all** of them was simply that they wouldn't have the time to do it. Even over a long weekend, there would likely be too many accounts in a large bank, and during normal business hours, a clerk trying such a thing would be noticed by the other clerks. Of course in a small enough bank, this breaks down.

On the other hand, if you wanted to find **all** of the accounts with balances between $5000 and $7500 to make them a special offer by physical search, good luck. Unless your file system was specifically set up to do the particular sort of function you wanted to do, it was too expensive to justify to have clerks go through **all** of the files to find **all** of the things of import. The solution might be to have clerks servicing those clients identify the situation as the client was present and notify them of the special offer at the time.

It seems that the lesson of the paper world is that inefficiency is beneficial to mitigating certain types of risk. Put another way, risk is aggregated in volume, and limited volume produces limited risk aggregation. But can we translate that into the computer world?

How we do things in computers

Just like paper systems, if the computer’s database is not set up to specifically cross reference by a particular thing of interest, the only way to find it is to search until found. And that means, at least potentially, searching **all** of the records, files, or what have you. With computers, we can very often search **all** because computers are fast at doing such things. In fact, they are so good at it, that we often search it **all** even though we could search part of it. This produces ANA failures.

As an example, suppose I am searching a database for the blue things that are also bigger than a bread box. And suppose further that there is a cross reference by color, but not by size. If I specify the search as (thing.color == blue) AND (thing.size == bigger-than-a-bread-box), the computer might (hopefully will) only check the size for things that are blue. But if I specify (thing.size == bigger-than-a-bread-box) AND (thing.color == blue), then every thing will be searched for size and only those that are bigger than a breadbox will be checked for color. In one case I search **all** of the records (an ANA failure), in the other, I don't.

Of course systems administrators don't normally search or change **all** of the things they can. If they suspect the problem is in a configuration file, they will look at the configuration files and not the database content files. I have had occasions when I had to update **all** of the files in a substantial collection to reflect a necessary change. For example, in updating a Web site during the pre- stylesheet and javascript days, I had to use a script to go through **all** the files. But since the last update, where I put the things that are common into a small number of locations referenced from the Web pages, I no longer have to do this. Of course the risk is now aggregated in a few files that, if changed, can have a large-scale effect. Thus an ANA failure was replaced with a different risk aggregation.

While there are a lot of other examples that show different aspects of the issue, I hope that some level of clarity is emerging with regard to the nature of this problem. We don't usually want access to **all** instead of **any**, but that's what we get a lot of the time. And when the individual responsible decides to abuse the access, bad things often happen.

What are the underlying roots of the problem?

This, it appears, is a problem worthy of attack, because it seems, for now, to keep fighting back.[1] If I had to try to track this problem back to root cause, I would guess it has something to do with mathematical models of protection and a long history of use. The mathematical symbol '∀′ (for all) is used to identify all, while the symbol '∃' (exists) is used to identify that there is something. Mathematics, in order to cover all possibilities, often has expressions like $Z:\forall x \in X, f(x)>3 \rightarrow x \in Z$, which expresses that, for all elements x of the set X, if a function 'f' on x produces a result greater than 3, then x is an element of the set Z. While there is no explicit method identified in the mathematics to determine membership of any real set, the most obvious algorithms to invoke such a thing runs through all of the elements of the real set X, looking to see whether they are greater than 3 by doing a comparison operation. It then puts all of the elements that exceed 3 into a new set called Z. If some elements of X are not numbers, then there is no comparison possible, and computers tend to produce exceptions, and so forth.

1 “Problems worth of attack, prove their worth by fighting back.” - Alan Perlis - painted on a wall at C-MU before the wall was altered for a new building.

For efficiency, when programmers or algorithm and system designers implement mechanisms to execute such mathematical concepts on real data sets, they tend to come up with better ways to store, search, and otherwise work with the data sets to improve efficiency and reduce the need to examine all elements of a set. But when doing systems administration and similar activities that are typically one-time and for exceptional conditions, it may take a lot more human time than computer time for the human to design, test, and implement a more efficient method for a particular search or other task than the time it takes to do it the brute force way. And not all programmers seek the elegance of better algorithms that minimize access to content. If it's good enough, use it, and fix it later if you need to – seems to be the most common approach. And they can get away with it, because they have access to **all** when they really need access to **some**. These ANA failures result from the lack of an ANA requirement.

How would we do ANA defenses?

There are historical examples of doing ANA defenses. For example:

- Rate limiting the number of things that can be seen, changed, or done per unit time is an ANA strategy. Thus **any** may be accessed over a given time period, but only the rate limited subsets of **all**. The question is whether rate limiting is too disabling for the specific purposes and whether the cost of ANA defense is too high for its value.
- Inference controls[2] that watch each access and prevent access when and if it could violate the aggregated inference constraints of the system have been around for a while, but they are rarely attempted, because they create other problems and are too complex and space consuming to accurately track.[3]
- Schemes in which several people must authorize an activity, but a larger group is available for such authorization, such as N of M voting schemes, may be used to authorize activities, and these involve **any** but not **all** of the authorizers.

But none of these really deal properly with risk aggregation issues. Still other approaches seem like possibilities. For example:

- Many systems threshold transactions, requiring additional authentication or authority at higher thresholds. A similar approach could be taken in which the sum of the values of transactions are kept for users, sessions, time periods, etc. and used to grant or deny further use without additional authentication and/or authorization.
- Structural disaggregation of risk can be implemented so that a distributed environment contains **all** of the relevant records but no individual has access to **any** of them. Rather, each human and mechanism is limited to a subset that does not exceed risk aggregation thresholds. They each have **any** or **all** access to their subset only.

Technically, such systems all seem to require that we either keep track of what has happened to determine what should happen next or separate out the totality of **all** into a set of smaller less risky subsets. Tracking means retaining state of some sort, and doing it in a trusted way so that the systems administrator cannot disable the function, alter its operation, or change the basis for its use. This means that when the systems administrator adds users to grant

2 See B. Fung, S. Fraser, and P. Yu, "Privacy-Preserving Data Publishing: A Survey on Recent Developments", ACM Computing Surveys, 2010 or 2011

3 See D. Denning, "Cryptography and Data Security", 1982, ISBN 0-201-10150-5

them access and then uses the newly created accounts to do the same thing, those users are also constrained so as to limit the systems administrator. Or perhaps multi-person controls should be required for functions that could violate the aggregated risk thresholds.

What are other well-known ANA failure examples?

Looking at risk aggregation, we have many examples of ANA failures.

- The WikiLeaks State Department memos case of a systems administrator leaking tens of thousands of classified memos falls in to the ANA failure arena.
- The leaks of large numbers of credit card accounts from processors is another example of ANA failures.
- Cases where someone alters a URL to turn a field value into a “*” and resulting in global release of records, are also ANA failures.
- Cases where programmers plant Trojan code to delete all records or files are ANA failures as well.
- The shell command 'rm -rf /' likely yields an ANA failure if performed from a privileged account.

What can we do from a practical standpoint today to mitigate ANA failures?

Current operating environments don't really support ANA defenses. There used to be various kinds of quotas on space and time, but they were never applied this way to my knowledge. Limiting bandwidth to systems is unlikely to be effective as an ANA defense because the bandwidth required for normal usage is high enough that the harm from ANA attacks is severe long before bandwidth issues cause problems. Some anomaly detection and response systems can handle a limited sort of ANA defense against traffic volumes, but this only effects content passing those detectors and only if they are tuned for such response.

On a theoretical basis, it appears at first glance that ANA problems cannot be resolved in a Turing capable machine. That seems to mean that limited content, limited flow, or limited function are the only obvious approaches to definitive resolution. But before we go too far down that route, we currently have problems even properly defining the the desired limits of ANA controls. And until we can define what we want, computers are unlikely to help us attain it. So we even have a definitional challenge that is yet unresolved.

Conclusions

It looks like there aren't any obvious ANA defenses we can simply throw up and expect to work. In fact, it appears that ANA defenses are a hard problem in information protection, are likely to remain so for some time to come, and that ANA exploitations are likely to continue for the foreseeable future.

ANA is just one of the many faces of risk aggregation that will get worse with time unless and until we come to understand the underlying issues and resolve them in a meaningful way. It appears that ANA failures are a direct effect of Turing capability combined with aggregation of content, and that there is no perfect resolution without limited function, limited information flow, or limited total available content.