All.Net Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

The assumption of breach[1]

It is a ridiculous notion that we should assume that everything is breached all at once. This can never work. The reason is simple. In such a system, reliability would be 0. You could never trust or rely on anything. So we need some sort of a trust model to make sense of the potential of breach, whatever that is. I prefer the term subversion[2] to breach to start. So I will use it to describe states (subverted / unsubverted) of a component[3] or a composite[4].

What might we reasonably assume?

Any complex system can be subverted. The reliability and availability equations from fault tolerant computing seem like a reasonable basis for creating a model of how subverted a system is over time. Here are the basics:

- Assume systems get subverted on a random basis with a mean time to subversion (failure) of F.
- Assume such systems get repaired (somehow unsubverted) with a mean time of R.
- The unsubverted portion of systems operation is then computed as (F/(R+F)), i.e., the time till failure (unsubverted time) divided by total time (unsubverted + subverted).

So systems that get subverted after a year of unsubverted service on average (F=1) and repaired after a quarter of a year of being subverted (R=0.25) are unsubverted F / (F + R), or 1/(1+.25) = 1/1.25=0.8 (80%) of the time. This assuming of course a simple sequence of unsubverted followed by subverted followed by unsubverted, etc.

However, in the reliability version of the world, failed parts are failed until fixed, and fixed parts are fixed till failure. But in security, we can have multiple simultaneous breaches. So even when we repair (unsubvert) one subversion, there may be other subversions in place.

If we take the example above, (R=0.25, F=1) but there are multiple components that this applies to, all of which have to be unsubverted in order to have the composite of those components to be unsubverted, things get problematic fast. A simple version is that for each component, there is 80% probability of unsubversion (20% chance of subversion at any given time), for a composite of N components, and assuming subversions are independent events, the probability of the system being unsubverted (i.e., all N components are unsubverted at the same time) is $P(P_1 \text{ and } P_2 \text{ and } \ldots P_N) = P(P_1) \times P(P_2) \ldots \times P(P_N)$, which for each component being the same $P \rightarrow P^N$, and for 80% unsubverted $\rightarrow 0.8^N$ so for 10 components, this comes to about 0.10737…, about an 11% chance of the composite being unsubverted at any given time. That means almost 90% of the time, the system would be subverted.

We can obviously apply this to more complex scenarios, as we should, using probability theory and identifying redundancy and interdependencies and measured times.

1 a gap in a wall, barrier, or defense, especially one made by an attacking army.
2 the undermining of the power and authority of an established system or institution.
3 a part or element of a larger whole, especially a part of a machine or vehicle.
4 a thing made up of several parts or elements (i.e., components).

Components vs. composites

I have been a bit unclear on this component/composite thing. Composites are composed of components, which themselves may be composites (they almost always are). The thing we are interested in is the properties of composites based on the properties of their components.

As an example, in fault tolerant computing, we can use redundancy to cover faults, so that a redundant voting scheme with 3 inputs and 3 outputs doing a majority vote on the outcome of one bit, only fails to produce the right output when 2 of 3 outputs are wrong values. Individual faults by components do not produce failures of the composite. Rather, at least 2 faults must be present to produce a failure. The faults may be in the components or the inputs. Such composites and their components operate over defined operating ranges of physical properties (e.g., temperature, pressure, voltages, currents, rates of change, fan-in, fan-out, induced flux levels, gravitational forces, impulse behavior, etc.), and if those properties are not held true, the predictions of the engineering analysis may also not hold true.

We can also build composites out of software components, which is how essentially all commonly used software systems are built. In most cases, the software is composed of components of software from multiple sources, each built from other components, etc. This is described as a supply chain, or interdependencies.

A subversion of a component in the supply chain can be carried through to the transitive closure of interdependencies, the mean time to subversion is unknown but studies have shown it to be very small when intentionally induced by authors of components, while the mean time to unsubversion is often in the range of years. The current capacity to identify interdependencies for unsubversion processes relies on the software bill of materials (SBOM) technology which is reducing unsubversion time and cost dramatically after detection of a subversion.

For a composite such as they typical user computer today, there are perhaps 10,000 software components at 2 levels of transitivity of supply chain. The time to subvert a component is likely hours to days when targeted maliciously, and really for many of these components, it's just a matter of resources applied to perform subversion. There are likely 100+ governments and hundreds of non-state actors at any given time performing such subversions, so even if there was only one such subversion a day on average for each of 100 actors, the current lack of change control on the supply chain leads to many known cases of months to detect. Let's be generous and say 10 days to detect and deploy a patch (i.e., unsubvert).

For any one component, the odds of subversion are then 100/10,000 = 0.01 subversions per component per day, or 100 days for F, while unsubversion (R) is 10 days. So then we will have F / (F + R) for F=100 days and R=10 days, or 100/110 = 0.909... (91%) unsubverted time and 9% subverted time per component. For 90.909…% unsubverted per component and 10,000 components, we get $0.9090909...^{10,000}$ for the unsubverted portion of the time for the composite, which is too small to calculate on my calculator. For only 1,000 components, this comes to something like $4*10^{-2040}$, again according to my (I am now convinced) almost certainly always subverted cell phone calculator.

The situation looks pretty grim up to here, and the assumption of breach is looking pretty good. Even when a small portion of components are subverted, composites relying on all those components are almost certainly subverted all the time.

It looks bad from here, however

I may have forgotten to mention that subversion of components and thus the composite does not mean that the composite is doing anything particularly harmful.

In fault tolerant computing, systems may have many faults that are never exposed and thus never produce failures. Other faults may rarely produce failures, and of course failures of a composite might not produce failures of a larger composite of which the failure prone composite is a component. Furthermore, some failures are worse than others. For example, if my calculator had a bad last bit in its calculations and calculated to 20 digits only showing 18, it would be rare when the last bit fault would produce an output that was not accurate to the precision shown, and if/when it was inaccurate, it would only reflect a minimal difference between the right answer and the presented answer. Indeed all digital calculators are inherently wrong for cases like the calculation of 1/3 (e.g., 0.33333...), where they can never produce a value that is fully precise and accurate, because no matter how many digits of accuracy you produce, the 3s never end.

As an example, when the log4j subversion was announced, I checked my systems, and found that something like 1 in 5 had an apparent log4j library installed. So naturally, I did the chmod to remove root privileges. But I also checked to see whether the subversion was exposed in my systems, and found that it was not. In my case, I tend to control the software operating on most of my platforms, and as such, none of my platforms were using software that ever used the library in question. Thus the subversion was present, but could never be exercised. Why then did I do the chmod? It was easy and in case I was mistaken or some software in the future might expose the subversion, I figured why take the chance?

There are two things involved here; (1) exposure of subversions to exploitation and (2) consequences associated with such exposure. Naturally, I will handle them in the reverse order here:

- **Consequences of exposed subversions are contextual:** The same subversion may be exposed leading to global thermonuclear war or exposed leading to an undisplayed last bit of a calculation for this article. Hopefully this is not true in reality (I suspect it is not but if I knew I could not tell you), but conceptually it could be true, and it is true for many subversions.
- **Exposure is a function of how the components are composed:** The same components composed in different ways produce different exposures of the same subversions. The mere presence of a subversion doesn't mean it will ever be exercised.

And then there is the HR component

I think I forgot to mention HR (Human Reliability) in my equations above. Of course I didn't really forget it, and the equations take it into account, but in order to use the equations properly, you need to include people as components of the composite for systems where humans are involved.

A longstanding widely used approximation is that 1/3 of people will violate employer trust regardless of situation, 1/3 will never violate such trust regardless of situation, and 1/3 will violate that trust under stresses of various sorts. Note that we cannot be precise here…

So, for an entity (composite) with 90 workers (the precision returns), 30 will always be subverted (time to subvert=0, time to unsubvert=5 years or whatever the mean time to termination is in the company → F / (F + R) for F=0 → 0% of the time unsubverted), 30 will never be subverted (100% unsubverted), and 30 may or may not be subverted and may change with time. If all 90 must be non-subverted in order for the composite to properly function, sell the company… which is why we have things like...

Countermeasures

The structure of a composite can be architected to reduce the effect of subversions. An example of this is separation of duties. By architecting the composite so that no component can create an obligation and meet it, the exercising of a subversion requires that both components be subverted, although not necessarily over the same time spans, in order for the composite to be subverted in this regard.

The time span issue brings us to the time transitivity of information flows and related issues that I will largely ignore for now, but in general has to do with the fact that composites are normally composed of sequential mechanisms and thus the composite is sequential and subversions of sequencing and sequential changes to the composite come into play.

From a countermeasure perspective, sequencing methods such as submit/commit cycles in composites allow for fail safe conditions in case of inadequate subversions, time limitations on sequential subversions, and other similar approaches. For example, a submit commit cycle assuming diligence by each of the two process steps and independence of the submission and committal, lead to subversions only producing composite subversion if more than one component is subverted or the mechanisms of composite sequencing are subverted. In the simplest case, if two mechanism must be simultaneously subverted, the equations for subversion change from any (e.g., $P(P_1$ and P_2 and … $P_N) = P(P_1) \times P(P_2)$ … $\times P(P_N)$) to all, the same equation, but for subverted components rather than unsubverted components, thus $P((1-P_1)$ and $(1-P_2)$ … $(1-P_N)) = 1-P(1-P_1) \times 1-P(P_2)$ … $\times 1-P(P_N)$), or in other words, for (F=1) and (R=0.25) 1-0.8 subverted (20%) of the time, and $P(P_1$ and $P_2)$ is 0.2*0.2 = 0.04 or 4% of the time both are subverted. Going to an N-step approval process brings $(1-P)^N$ for all N subverted, and we can drop the likelihood of subversion to any desired degree by redundancy, remembering that this redundancy brings with it the assumptions no collusion (i.e., independence) and alternative methods of subversion.

For example, suppose we wish to deny services to the composite rather than cause it to produce a wrong result? Remember that we were generic with regard to the concept of subversion. Multiple steps in a sequence increases the number of non-subverted components required in order to perform the function of the composite. Increasing the sequence length also increases the time required for the composite to produce results. We are subverting the performance of the composite by architecting it to reduce the effect of subversions through sequencing. In a complex composite with timing and reliability constraints around communication between components, subversion of commonly used components, such as communications or mechanisms sharing the same components (e.g., a DNS server, a network, etc.), we have higher valued targets for subversion because the composite is not equally divided in its dependency on the components. A disease spreading among the people may cause delays in processing commits, leading to timeouts in processes, forming a complex interdependency chain of subversions to the composite.

The unevenness of subversions

Some people will be loyal, some will not, and some may or may not be depending on the situation. That is the nature of things based on the history of people and their loyalties. The problem in terms of metrics is identifying which is which. The problem in terms of composite design is measuring the components so you can get the right mix of redundancies so the composite comes out with the desired level of overall subversion.

The Defense Personnel and Security Research Center (PERSEREC)[5] is a Department of Defense entity dedicated to improving the effectiveness, efficiency, and fairness of DoD personnel suitability, security, and reliability systems. They have done this over the years by studying the behaviors of people, including turning behaviors, and ultimately finding ways to better adjudicate background investigation results. This field of endeavor has largely become automated because of the vast amount of available information on people and the historical findings about loyalties and how and why they happen. But human reporting (sensors) of indicators[6] remain one of the most important ways that subversions are identified.

The time frames for detection and mitigation of human subversions tend to be quite long, and in the detected cases where the worst harm has been done, the failure to systematically and in a timely fashion act on detected subversion behaviors has been a common theme. The same has held true for major losses associated with software subversions, where automation has made mitigation timeliness far more important and time frames before harm far shorter.

Then we have the issue of collusion. In many cases, multiple actors collude to subvert a composite. Outsiders acting with insiders historically produced something like 2/3 of the losses in the cyber realm. And if you look at examples like cyber extortion, there is almost always an inside and outside component involved, albeit the inside component is most often not intentional. On the other hand, a group of something like 5 employees at a branch of a major global bank colluded to steal from the bank in recent years. At the end of the day, if enough components of the composite collude (or combine in parallel and/or in sequence) to subvert the composite, the composite will be subverted. Thus combinations and sequences of subversions must be analyzed in order to gain clarity around complex subversions.

Some bounds on subversion

Building a model of subversions is problematic, in no small part because of this wide unevenness. Simple models lead to simplistic conclusions. However, we might be able to put some numbers around bounds on subversions.

One approach is to take the common vulnerability (CVE) database and identify the number of known subversions of any given operating environment over a time frame to get a lower bound on the rate of subversions. We can then try for a lower bound on time to mitigate from published studies, of which there are several, or use the “patch Tuesday” approach of at most one mitigation per month (in most cases) and thus an average of 15 days till mitigation (as a lower bound). These are obviously best case assumptions, but... For Windows 10, we get about 50 new CVE entries per month[7] For F=0.6 and R=15 we get F/(F+R) = 0.6/15.6 = 0.04, or a 4% change of each Windows 10 system being unsubverted at any given time.

5 https://www.dhra.mil/perserec/
6 https://www.dhra.mil/Portals/52/Documents/perserec/reports/core_brochure.pdf
7 https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-32238/Microsoft-Windows-10.html

Back to breach[8]

There is another similar meaning of breach, which is, in essence, the act of breaking a deal. These acts are then associated with a threat actor. Which brings us to TxV→C (threats exploiting vulnerabilities produce consequences) and the sequential nature of attack and defense.

While we can reasonably presume that widely used composites such as Windows 10 are subverted essentially all the time, and that any substantial network is always subverted as a composite, that does not imply that any of the composites or components of any given composite have been breached in this sense of the word.

The assumption of breach presumes that for any given composite of interest, it is currently in a state of breach. That is to say, that in addition to being subverted, subversions are being exploited to produce consequences. Or in the TxV→C world, that threats have or are exploited or exploiting vulnerabilities to produce consequences.

It's important to note that consequences occur over time, so that if all the consequences have happened, there is nothing you can do about them, and it is a waste of time to try. In effect, past consequences have already occurred, and from a breach standpoint, there is nothing we can do about it, even if there may be post-hoc mitigations.

We know from the analysis of subversions that vulnerabilities are very likely to exist in substantial composites, and certainly in commonly used operating environments. Assumption of vulnerabilities, however, does not produce the conclusion of assumption of breach. We still have threats exploiting those vulnerabilities to produce consequences.

As previously discussed, in order to exploit a vulnerability (cause a subversion to produce an effect), proper inputs must be placed in the proper locations in the proper sequence and resulting outputs must occur in the proper sequence and places to produce consequences. This is essentially the counterargument to everything being breached all the time.

Countermeasures

Every usable system has countermeasures to breach. The inherent redundancy of physics in the electronics of components of computer systems is produced by not having each logical component comprised of a single molecule or a single electrical or magnetic charge. Rather, multiple molecules and charges are used for storage, communication, and transformations in the finite state automata that comprise circuit composites. The subversion of these mechanisms at the atomic level is constantly present, yet we build reliable systems from unreliable components every day. Other forms of redundancy at the hardware level are common.

As we get to higher levels of complexity both in terms of components and composites, we end up in more complex architectural structures where there innumerably large numbers of subversions possible relative to the innumerable number of different aspects of behaviors of the composites. Even the definition of "secure"[9] has many different interpretations and facets when it comes to all possible subversions.

8 an act of breaking or failing to observe a law, agreement, or code of conduct.

9 Note that the term secure is not just possibly a noun. It is a verb [fix or attach (something) firmly so that it cannot be moved or lost], and adjective [fixed or fastened so as not to give way, become loose, or be lost] and as often used in the cybersecurity arena, a state of being [free from danger OR according safety, etc.]

Countermeasures can never produce a composite that is in fact secure by the “free from danger” definition, nor can there be any such composite, and of course the term “danger”[10] is itself not informative as to specifics. The consequences of “injury, pain, harm, or loss” are ever-present and can come in many forms. Arguably, any effort we spend to secure a composite produces harm in the loss associated with the effort, and thus the cost vs. loss tradeoff comes into play. Furthermore, the mechanisms of composites are subject to all manner of variances within specifications, and even the specifications of components are rarely in terms of the various possible event sequences of components that can cause the composite to produce composite loss. In simple terms, we do not even understand an underlying science of what we mean by security.

But we do have countermeasures to historically identified and projected future event sequences with known ranges of consequences. And there are lots of them in place in almost every current commercial system used as a component of a networked environment. And the architecture of those environments include many components that cover many of the paths from threat to consequence.

This ultimately leads to issues of attack graphs which have been explored for some time, and this leads to the notion of modeling and simulation, also explored for some time, which leads to the notion of model-based situation anticipation and constraint[11] [12] [13], a generalization reflective of how thoughtful decisions are made. A list of areas of countermeasures and attack methods at a level of granularity suited to simulation and related analysis from some time ago is available for those interested.[14] [15] Issues of granularity lead to problems in modeling and simulation complexity, limiting the use of high-end methods to entities with a lot of resources, however, simplified versions are used for risk management based on probabilistic or other models.[16]

One final problem is that instrumentation is inadequate to identify all breaches, even after they take place. Thus any estimates based on facts will at best be lower bounds on actual breaches in place. As an example of this, recent estimates of “exploitation events” having reached 48,904,064 in Q1 and Q2 of 2022 might lead to some analysis, except of course the lack of references for the claim and no information on what this may actually mean in terms of consequences or systems or anything else is problematic. A more reliable source might be the 2022 breach investigation report from Verizon[17] which indicates as a headline “23,000 incidents and 5,200 confirmed breaches from around the world”. That from their subset of actual activities.

Fear Uncertainty and Doubt dominate the available information, and very few of the so-called studies lead to information that could reasonably be used for an actual analysis of breaches. While these fear-inducing mechanisms may be good for selling cyber-security, they do not enlighten us as to the real nature of breaches or the assumption of breach.

10 exposure or liability to injury, pain, harm, or loss
11 http://all.net/Analyst/2021-07-20-Forefront-Cyber-Risks-Nuclear-Safety.pdf
12 http://all.net/journal/ntb/cause-and-effect.html
13 https://youtu.be/3Q1w6a9241g
14 http://exec.all.net/game?what=Responder
15 http://all.net/journal/ntb/simulate/simulate.html
16 http://all.net/SoP/SecDec/RM0.html
17 https://www.verizon.com/business/resources/reports/dbir/

Realities

In reality, all systems are not always breached, in the sense of producing identified losses (negative consequences). As an example, I run a server at Amazon Web Services (AWS) that continues to provide its utility over time. The nature of the system is that providing that identified utility is the only consequence of import to me, the “owner”, and as such, a breach would have to change this in order to be counted.

Indeed there are billions of computers all over the world providing utility to their users all the time, including billions of cell phones, hundreds of millions of networking devices and servers, and lots of other similar sorts of composites.

Furthermore, a breach without consequence (actual harm) is likely irrelevant.

> If a tree falls in a forest, does it make a sound?
>
> It depends on your definition of sound.

People spit on the street, which is against the law in many places. Researchers violate codes of conduct in the cyber-security field all the time, and mostly they don’t even know they are doing so. Most online click agreements are so complex as to be impossible to fully understand, and thus are likely violated all the time. Each one is a breach by the definition, and the vast majority of them have no identified consequence.

What does the assumption of breach get us?

Some seem to claim that this is about awareness. One we assume breach perhaps we are obligated to act to mitigate. But mitigate what? We know by now that we cannot hope to mitigate all of the subversions of systems, and we cannot even identify many of the breaches that we already believe are occurring. We cannot perfectly prevent these breaches by any known methods, at least not without removing the utility of the systems we depend on.

The assumption of breach today is being used to sell more security stuff, and one of the major negative consequences is the increase in security load.[18] [19] It is truly unclear whether this increased load combined with the increased out of pocket cost is worth any protective value gained. It is even less clear whether the solutions being offered in this regard and under this banner offer effective improvements compared to alternatives available for mitigation.

As a general rule, you can defeat any system by identifying the assumptions and violating them.[20] The assumption of breach is no exception. When you apply this approach, you exhaust resources that might otherwise be spent on other things, like succeeding in your enterprise. And as a competitor, I like your waste. Efficient use of resources is key to my competitive advantage.

Conclusions

We know that systems including human and non-human components are subverted and that subversion does not always lead to breach. We know that assuming breach is used to apply fear, uncertainty, and doubt to cause customers to expend resources on defenses, but also that it provides no guidance as to mitigation. In summary, it is a useless assumption.

18 http://all.net/Analyst/2021-06C.pdf
19 http://all.net/SoP/SecDec/HumanLoad.html
20 http://all.net/Analyst/2021-03C.pdf

