Obsidian: A Safer Blockchain Programming Language

featuring Eliezer Kanal and Michael Coblenz as Interviewed by Suzanne Miller

---

Suzanne Miller: Welcome to the SEI Podcast Series, a production of Carnegie Melon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the US Department of Defense. Today's podcast will be available at the SEI website at sei.cmu.edu/podcasts.

My name is Suzanne Miller. I am a principal researcher here at the SEI in the Agile-in-Government program. I am very pleased today to introduce you to two of my colleagues Eliezer Kanal and Michael Coblenz. They both work on a language called Obsidian. Eliezer "Elli" is a technical manager and researcher in our Cyber Security Foundations [group] in our CERT Division at SEI. Michael is a doctoral student studying programming languages in Carnegie Mellon's School of Computer Science. So welcome to both of you.

Eliezer Kanal: Thank you.

Suzanne Miller: We are here today to talk about a new programming language called Obsidian. Before we do that, we want to hear a little bit about the two of you and what brought you to this kind of work. Developing new programming languages is not in the, G*ee when I want to grow up I want to do a new programming language*, kind of dreams of people. So how did that happen?

Eliezer: The work that I do actually at Carnegie Mellon University mostly relates to data science. I am actually a technical manager of a team of seven including myself. Most of the projects that we do revolve around the use of statistics in light of various types of security issues. So insider threat, network defense, malware analysis, vulnerability analysis all that sort of things, applying these technologies there. When blockchain programming, blockchain technology started becoming much more popular, we ended up just getting into that because it was a hot topic, and it was something that people needed to pay attention to. Language security turned out very quickly to be a real issues in that area.

Suzanne: Michael, what about you?

Michael Coblenz: I used to work in the software industry. I was in Silicon Valley for a long time. I noticed that there were all these debates about how we should be designing and using program languages. Nobody could agree about what language features were a good idea, what languages were a bad idea, or how to use program languages in an effective way. I became interested in doing research on program language design so that we could better understand particular language technologies that would be helpful in making software more reliable and making software engineers more productive and also design methodologies that language centers could use to achieve their goals more concretely, more successfully.

Suzanne: And so the marriage became the Obsidian project. Before we get into that language specifically, why don't you give our listeners a little bit about why is it important that blockchain as a technology has some languages that reflect its usage and reflect the needs of the programmers? So give us a little bit of a mini-tutorial on blockchain.

Eliezer: Blockchain, very broadly speaking, is a type of data actually if you think about it at a pretty fundamentally. It is a way of storing data where the data itself does not change over time. In fact, it can't change over time and only add to it. In addition it has a couple other useful properties such as there is an identity automatically associated with it. How you get that identity is a topic, but it has an identity automatically associated with it. It is very easy to be audited. It is distributed processing, so it is not being all run on a central server, which makes it very resilient. So there is a lot of interesting properties that this new technology has.

Over time, people have been investigating and try to experiment, *how do you use this technology*? The early versions of the programming languages that were made available so that people can work with this technology ended up having a lot of problems. There were some very famous ones. Probably the most well-known is a piece of technology called the DAO [decentralized autonomous organization].

When the DAO came out, there was a lot of people interested in it. Very briefly it was a venture-capital-type group where people could put money into this thing and then, through voting on a blockchain, send the money to a specific party. Because of a programming error the DAO lost tens of millions of dollars. Someone just siphoned it right out of the organization. It became very apparent that the programming error was of course the fault of a programmer, but it was also the fault of the programming language, because the language was very hard to develop. It is very easy to make mistakes in it. So as we are looking at this, we are trying to figure out, *How would you design the language that makes more sense?* That is exactly where Michael works.

Suzanne: The primary use that most people will know blockchain in relationship to is a currency called Bitcoin. So that may be where if they don't know blockchain people may know Bitcoin as the latest thing in cryptocurrency. This is where the identity and the resiliency and all those kinds of things of blockchain make that a technology that is really useful for Bitcoin. But we are also seeing it, there are recent things in the news and what not about it being used specifically for high value encrypted data as another sort of place that blockchain type of technology seems to be going. The kind of language issues that you are talking about anytime I want really well secured data, I am going to have the issue of, *Can the programming language enable that*? *Or does it make it more difficult to do that*?

Eliezer: To jump on a point that you just said, for those for those who aren't familiar, Bitcoin is a blockchain, but not all blockchains are Bitcoins. It is very important to recognize, Bitcoin is a financial instrument at this point, and you can invest in it, and you can make money with it and all that sort of stuff. Not all blockchains are going to require that.

In fact we are seeing many chains who are opting to turn themselves into a financial instrument as well using these ICO's and currency offerings and everything like that. Not all chains require that either. In fact you can have private blockchains where it is run just like any other enterprise application on your own network without any sort of public coin offering. It is important to be aware that not everything is a blockchain.

Suzanne: This is an underlying technology.

Eliezer: Exactly.

Suzanne: Not an end user application.

Eliezer: Exactly.

Suzanne Miller: That is the other reason that having languages that support the different uses of blockchain is important.

Michael: One way of thinking about Bitcoin is that it is actually fairly limited in that what it tracks is account balances. Every user has an account balance. Each individual person may have many different accounts. But the point is that it tracks account balances. Whereas other blockchains, Ethereum for example, are actually much more sophisticated in terms of tracking entire programs. Instead of just saying, *Here is an account, and here is how much it is* you can actually install programs on the blockchain. So when you write a program then you have to answer the question, *What program language are you going to use to write this program*?

Suzanne Miller: *And how do I make sure it's safe? And how do I make sure it's secure? And how do I make sure that it does what it says and only what it says?* Those are kind of the three things.

Michael: Right. *And how do I ensure that it does what I want it to do*? *And not something else*?

Eliezer: It might do what it says it is going to do, but if I don't want it doing what it says it is going to do, I would be in trouble.

Suzanne Miller: There is that. That's true. There is a lot of interest in the defense community, the Defense Advanced Research Projects Agency DARPA, others. The transparency/resiliency/forgery resistance when you start talking about currencies are all important. How are you seeing blockchains being used in the government settings, not just in the financial market, but other government settings?

Eliezer: I just got back from a blockchain symposium in D.C. actually. At this point there are a lot of government agencies investigating the use of blockchain. The applications range from financial management, where they are trying to manage the NIPA [National Income and Product Accounts] system. There was an SBIR [Small Business Innovation Research] going out to help with that.

Suzanne: That is the system for transferring money among government agencies.

Eliezer: Thank you. There was also an application being looked at, many applications for managing supply chain. The audit ability of blockchain makes it that when pieces enter a supply chain you can easily track it after the fact, even multiple layers deep in the supply chains. Your supplier bought from this supplier, bought from that supplier. Usually you would not know that, but if all that information is on the blockchain you will see it.

We are also seeing a lot of applications in simple standard automation where people are taking standard processes…One of the well-touted ones is the GSA. They took their standard process for on-boarding new government contractors. They put a component to that on the blockchain and released it. That is one of the first real government applications of a blockchain technology. When you are talking to different people you are seeing it being investigated everywhere from finance to healthcare to military to communications. It is really almost all areas.

Suzanne: OK. Alright. This is something that you might as well get used to because it is not going away.

Eliezer: Definitely everyone is investigating how to apply it. People seem to be approaching it with caution. There seems to have been a lot of lessons learned from previous attempts to bring

new technology to the government. People are being let us put it cautiously optimistic. They are watching it very closely and watching the people who are trying it out and seeing where it goes.

Suzanne: So let's talk about Obsidian itself. The whole point of this podcast is to help people understand what is Obsidian? How does it do the things we are talking about? What made you think that there needed to be a language other than the languages that are already out there for blockchain. Obviously the one example you gave from DAO is one reason, but are there other catalysts for a new language entirely?

Michael: There are several different observations we made that started this project off. In addition to the DAO theft there, there have been others. The observation is that actually people have a really hard time writing correct programs in the languages that people are currently using. There are two approaches that people are using right now. One approach is this language Solidity, which has been shown to be insecure by the examples we were just discussing. Other blockchain platforms are using existing standard languages like Go or Java.

There is certainly plenty of bugs in Go and Java programs also. We looked at the set of application domains that people are interested in using for the blockchain, and we observed some commonalities. We were interested in whether we could actually design a language to provide stronger guarantees than you would get if you were to program these same applications in Go or Java. One observation is that a lot of these applications are typically stateful at a high level. To take one example think of a bond financial insurance. A company issues a bond available for sale, and then eventually somebody can buy the bond.

Suzanne: It changes state from being available for sale to being purchased.

Michael: That's right, exactly. As soon as the bond has been purchased it does not make sense to purchase it again. It has already been purchased once. If you were to implement this in a standard object-oriented language like Java, you would have a method that allows any caller to buy the bond, and it would do some sort of run-time check to check to see whether the state was valid for the transaction that was being evoked. That's OK, but it relies on a dynamic check.

We were wondering whether we could actually provide that kind of guarantee at compile time rather than a run time. By moving the guarantee back earlier, we could catch the bug much earlier. It has been shown that it is cheaper to fix bugs if they are discovered earlier rather than later. But in the context of the blockchain, this is especially important because the blockchain programs are immutable. Once a buggy program is published on the blockchain, it can't be fixed directly. One has to actually replace the program with a new program and then convince all of the users of that program to migrate to the new version.

Suzanne: That is one of the attributes of blockchain is that permanence. *Once I have been added to a blockchain, I am in that spot more or less forever unless I get replaced or added to.*

Michael: Right. There is a technology called typestate that we are using to represent high-level states of software interfaces, and let us reason statically—that is, at compile time—about the states that are available and what operations are available in each state. So we can provide compile time guarantees about that kind of thing.

The other observation we made is that a lot of these contracts, these programs, are based on some sort of resources. Think of money for example, a virtual currency or some other kind of resource. These quantities are actually very different from the kinds of objects that you might typically pass around in object-oriented programming languages. For example, if I construct a color object to represent the color of a pixel on the screen, and I pass it to some interface. Well, we will both have a reference to that color object. If I modify it, well we maybe should have established some kind of agreement regarding what the modifications were going to be thus I am violating anybody's invariants.

Suzanne: Like a service level agreement among, among different parties.

Michael: Yes, or some sort of API [application programming interface] specification. But this is actually very different from money where if I give you money, well I don't have money anymore. If I give you money it is not my money anymore, now it is your money. I can talk about how much money it was, but I cannot give it to you and then also give it to Eliezer. Because I already gave it away once. I cannot give it away again.

So similar to thinking about the types it guarantees, you could try to track this sort of thing dynamically and that's fine, but we want to actually catch these kinds of bugs earlier. You could imagine a bug in which I accidently give away money twice. You could also imagine a bug in which I…

Suzanne: Couldn't get away with that one.

Michael: You could also imagine a bug in which I take some money and then actually forget to do anything with it. I assign it to a variable, which is in some scope. Then there is a bug in which that variable goes out of scope, and that reference to the money is lost.

Suzanne: That would not make me happy.

Michael: No, because the money is just gone. You can't get the money back. It turns out there is a lot of different quantities in blockchain applications that have this kind of property. Think of votes in a voting system. If I have some sort of organization, and I want to let you vote, then maybe I give you a special token that you can use to vote. You don't want to accidently lose the

token, and you don't want to use it more than once. Maybe you can give it to somebody, but you can't give it to more than one person. It acts exactly like money in the sense that it acts like a resource. You do not want to lose it, and you do not want to accidently duplicate it.

Suzanne: Basically anytime you have a static resource pool where the change of state of the resource actually changes where essentially the resource is located, for lack of a better term. You want to be able to guarantee that that transaction can actually occur. That it's appropriate, and that it can occur, and that it actually has occurred before you actually get to run time. Because if you get to run time, if you took my money and that variable went away, I'm already in trouble. Right? And very upset with you.

Michael: Right. The term we are using in Obsidian to think about this concept is *ownership*. So I own some money, you don't own any money yet. If I give you the money now you own it, and I don't.

Eliezer: From the standpoint of the user, I am not a programming language researcher, but I have used the language now as part of their studies to figure out how to make the language look. It makes it much easier to track what's going on. So if you are writing a regular program, and you are trying to track all these things, you're doing it manually. I have to keep counts to make sure if I spent all the money. It is up to me to make sure I didn't forget to spend it. It is up to me to make sure I don't try to spend it twice because it's just a number. When you are working with Obsidian, if I try to spend it twice, as soon as I hit the save button so to speak, it warns me you're doing something you can't do. You're spending it twice. You forgot it or whatever it is, and it's much easier to work with.

Additionally from the state standpoint, if I try to spend the money before I got it, it will also warn me, *You are in an area where you have money to spend*. It makes it much easier to find the kinds of bugs that have plagued all these other users previously. One bug we didn't mention—Michael mentioned the possibility of money being lost—there was actually an issue recently on one of the larger exchanges of Ethereum, I believe it was Ethereum, that there was hundreds of millions of dollars lost because the coins were suddenly put in this part of a contract that was not accessible. Simply because the guy wrote the contract in a certain way, you could no longer pull it out. It was not exactly this issue, but it is possible to lose money through these types of problems.

Michael: In addition to reason about these things in a type oriented way, in the future, we would like to think about things to be static analysis. So we could actually write static analysis. There is code that runs in the complier to check to see whether there is any way that money can actually get stuck permanently in the contract.

Suzanne: I never even thought of that issue of the money getting stuck in the contract inside of the block. That is the kind of thing that a language, if it's not aware of that possibility... And that's really a lot of what you're talking about is, is this is what we would I would call a domain-specific language, right? It is not a language you are going to use to program a video game. It is a language specifically for this purpose, and it's taking advantage of the attributes of the kinds of uses that this the blockchains are going to have potentially and that we already know they are going to have.

Eliezer: So specific to that point one of the issues and the reasons that we are having so many bugs is people programming these languages right now are coming from web-development backgrounds, game-development backgrounds, finance development backgrounds. But they are not used to thinking in terms of *I have money and I have to treat it differently*. The correctness of their program is currently up to their creativity and their ability to recognize all the edge cases. We are trying to make a language embed some of that in there. Even if they didn't catch it themselves, they did not think, *Oh what happens if this happens*?

Suzanne Miller: Because the chances of an accountant that understands all these things, learning Java and all those things and becoming proficient at it and being able to marry that is, relatively small. There are some, but it is more the case they are going to hire a professional programmer who doesn't have the depth of domain knowledge. So you are embedding some of that domain knowledge into the constructs of the language.

Michael: The current standard practice for people programming in Solidity for example is that what people do is they say, *Well, if there's some error condition that has happened in this contract, then there should be some escape hatch. So there should be some trusted third party who we trust with all the resources in this contract and then we are going to just transfer all the money to that person because there's some error condition that has happened that was unanticipated by the author this code.*

Suzann: This is better than losing the money all together.

Michael: Right. So it is better than losing money altogether if you have a trusted third party, and that trusted third part actually acts in a trustworthy way, and the author of the code actually remembers to insert this trap door in every appropriate place.

Suzanne Miller: That's a lot of ifs.

Michael: It's a lot of ifs.

Suzanne: That is really what you are trying to prevent is needing all of those ifs every time somebody tries to make use of blockchain.

Michael: Right. In general trying to catch bugs earlier complied time rather than waiting until the system is deployed by which point it's too late to fix them.

Suzanne: I have to guess that there are programmers out there that are going *Okay I need Obsidian. How do I get Obsidian*? *Where is it in terms of its transition? How do people get a hold of it*? What are, what is its state of, of availability for programmers who want to use it to make safer blockchain applications? How do they get a hold of it?

Michael: Great question. Obsidian is a research project. It is not really ready for use by users of blockchain systems quite yet. One of the other research objectives with Obsidian is actually to think about how we should be designing languages in general. In terms of not just producing a language and informing people that this is what they should be doing, but actually trying to justify the benefits of the language and justify the design process that we are using to design the language in a scientific kind of way.

This makes language development perhaps a little slower than in a traditional context because we are trying to be a little bit methodical about how we actually design the language. People are welcome to take a look. We have a website its tinyurl.com/cmuobsidian that people can take a look at, but in terms of actually writing blockchain software right now Obsidian's probably not ready for use.

Suzanne: I would imagine that one of the things you are interested in, in understanding the breadth of application for Obsidian is if people say, *You know I have tried to do this kind of blockchain application, and solidity and these other languages just didn't work*. I would expect you would be interested in what are some of the boundary and edge cases for Obsidian itself that we may not have thought of here at the SEI? That kind of information would be useful to you from the community.

Michael: That is right. We are especially interested in particular use cases that people are working on, situations where they are writing some contract, and they are interested in whether Obsidian might a better way of writing their software.

Suzanne: Or here is the problem I am having in trying to create a robust blockchain application, and that may inform some typing constructs and other things that Obsidian hasn't dealt with yet.

Eliezer: Another way we have been reaching to the community has been trying to figure out, you have existing blockchain applications, or even if you have existing prototype blockchain code, what kind of bugs are you finding early on? Because we are trying to make sure that we are catching as many bugs that can be coded as possible. The more code we can see that has bugs in it—which basically means any code around because every code has some bugs—the more of that kind of code that we can see, the better we can develop Obsidian to be preventing those bugs as

well. We have been reaching out to a couple different agencies, a couple different corporations in trying to get as much of their code as we can with appropriate agreements in place, to make sure we have visibility into all the different types of errors that can be introduced.

Suzanne: Do you have any sense of when you think Obsidian will be a publically available language?

Michael: That is a good question. I don't want to make any promises because it is still a research project. Let me know when you can fund an engineering team.

Suzanne: Always back to the money. I can see that you have been learning the blockchain/bitcoin ecosystem well. Show me the money. Well that's fair. People need to understand this is early on but, but I think the fact that there is promise for a language that may help people to do a better job of this and to avoid some of the problems that we have seen already is one of the things that may actually allow blockchain to survive because everyone is interested in it as you say. I do not think it is going away, but there are catastrophic cases where it could, and some of those could relate to programming issues. It is like we can't trust it, so we can't do it.

Eliezer: This is currently a research project funded directly by the SEI. We have here competitive grant program. This is actually being funded by that competitive grant program as a research effort. We would be more than welcome to talk to any government agency or any sponsor who is interested in seeing this research all the way through and helping us you know develop this into an actual…

Suzanne: Piloting it in.

Eliezer: Piloting it and using it as a project. That's it exactly.

Suzanne: From what you've seen with blockchain because I know you've looked at a lot deeper at it than most people, where do you see it going? I mean what are the limits that you see for blockchain? Not just because of possible programming issues, but just sort of the construct itself. What do you think are the things that are coming ahead in the next five years or so?

Eliezer: So it is kind of interesting. Right now what blockchain is doing, given that it's, the new kid on the block, it is making a lot of organizations look at their own internal processes. And what we are seeing especially at this summit that I went to and at a different one I went to earlier, there is an awful lot of automation efforts that are being uncovered. What is coming out is that not all those automation efforts are actually related to blockchain. But it's being used as a catalyst for a discussion, which I think is exceedingly healthy all by itself. People are looking at the technology and saying, *Maybe this will help*. They look into it and say, *No it probably won't,*

but I should do this anyway because I just found there's actually another solution to it, which is great.

As far as blockchain itself, the areas that it is going to win most are areas that take advantage of the core properties. The three examples that I think are most relevant: finance. In finance I have to have a conversation between uninterested parties, which is exactly what Bitcoin is. I need to have audit ability. This is exactly that sort of system. The fact that it is distributed in very resilient to attack makes it all the better.

There is another area which we mentioned before of supply chain. Supply chain seems to be an area where anyone needs to be able to look anywhere in the system at any time and generally speaking you are only appending to this system. You are only adding new things and introducing it in. The fact that it has identity associated is all the better. It can really benefit from all the core properties.

The last area, which is a much broader area, is contracts in general. A contract is an agreement between two uninterested parties where there is an identity associated then an audit ability requirement. That concept of contracts really ranges from movie tickets, to marriage licenses, to deeds on your house, to actual legal contracts. There is a lot of interesting work going on trying to figure out, *How do we create contract based economy using blockchain technology taking advantage of all the inherent benefits that it has?*

Suzanne: If I just pull one of those out when you go to get a mortgage, right now there is a lot of manual processes that involve navigating that trust gap, right?

Eliezer: Right.

Suzanne: *I need to provide you will all these documents and things like that*. One of the potential uses of blockchain is to automate some of that contracting by being able to locate and marry up the different elements that create that trust.

Eliezer: Even more than that, then the mortgage is public and we can demonstrate that we have the ownership rather than having to go to the bank and pull out the lawyer. We can easily transfer the ownership of the house by simply taking my mortgage and shoving it on over rather than getting a whole bunch of lawyers involved.

A lot of this comes back to standard automation for which frankly you don't need blockchain. It is good to start the discussion, but you can do automation in many, many ways. But because there is multi-party, and because we have identities associated, I can authenticate who you are. I can create a transaction, and it can be recorded for posterity in a way that is very, very difficult to manipulate.

Suzanne: Yes. So once we have a language that is safe and trusted to use, then many of these other applications where we would create very large automation kinds of efforts might be actually much easier, much simpler. So that is really the key there.

Eliezer: Exactly. The one that I didn't mention is healthcare. And healthcare is having a huge effort because right now if you go to see one doctor, and you go to see a second doctor. First of all, the first doctor doesn't see the records of the second doctor. Second of all, you may not see your records at all because of strange ownership issues. Third of all, the insurance company is going to have to do all sorts of weird maneuvering. Imagine if all this data was available on a blockchain where all you are doing is saying, *Add to my medical record. Add to my medical record, and then you can pass it on as necessary.*

Suzanne: Then I can it pass the Ownership. Or share ownership in that piece.

Eliezer: Share ownership is necessary. There are all sorts of complex scenarios that are available, but you are essentially sharing information between untrusted parties where the information is not going to be changed.

Michael: There is some interesting conversations going on in the blockchain community about exactly how it should be designing blockchains and what the application spaces are for blockchains to be appropriate. One particular example is the question of public verses private blockchains. Some people say, *Blockchains should be out in the open*. Think of Ethereum, for example. *The entire internet should be able to see everything that is on the blockchain. If you'd like to publish some private data to the blockchain then you need to encrypt it first*. But anybody can fundamentally access all the data on the blockchain and see what's going on. Other people say, *Well a lot of the applications that people are interested in actually are fairly identity-oriented and private. Think of finance or think of supply chains. Banks may not need everybody in the whole world to see everybody's account balances*. That's probably not how most banks think of things.

Suzanne: I really don't want everybody seeing my account balance most of the time.

Michael: Well you could imagine that your actual account balance is encrypted in the blockchain. But if you are going to encrypt everything is it really that useful to have it all in public? So there is that question going on in the community.

Another question is there is a tension between high throughput and high reliability. Think of a credit card transaction processing system where a large credit card system might have thousands of transactions per second going on around the world. Well that is a much higher bandwidth in terms of transactions then for example mortgages at a bank or healthcare records at one particular doctor's office. So depending on which kind of applications blockchains are designed

for, that will tend to effect the applications on it. Blockchains are very popular and so people tend to gravitate toward blockchains just because they have some…

Suzanne: It's the new thing.

Michael: It's the new thing. So I think people need to be very careful about adopting blockchains. They need to think about exactly about why a blockchain be helpful for this particular application. Is there something a blockchain offers that other technologies don't? Because otherwise a blockchain is a very computationally expansive way to do things.

Suzanne: Like everything else there is no silver bullet.

Michael: That's right.

Suzanne: But this has some interesting promise, and the fact that we are trying to actually build a language to take advantage of this domain's weaknesses and its strengths. We are trying ameliorate its weaknesses, and [the fact that] we are trying to take advantage of its strengths is a really a powerful thing so. I was really glad to get a chance to talk to you both about this today. I thank you for joining us.

Those of you that want to know more about blockchain and the SEI's work in this area and data security and all the kinds of things that we've been talking about, look at insights.sei.cmu.edu. And search on Eliezer K-A- N- A- L, and you should be able to find his work. He is very prolific.

So again, thank you both. Other information that you need from the SEI on this topic, please don't hesitate to contact us at info@sei.cmu.edu. Once again, the podcast will be on iTunes U as well as our website sei.cmu.edu/podcasts as well as on the SEI YouTube channel. Thank you for viewing.