8 – The Tectonic Component

(Formerly the InDoor Operating System)

Question 8 *How do we bring the benefits of InDoor spaces to our computers, tablets, and phones?*
Answer 8 The InDoor Operating System
We can work around the vulnerabilities of popular operating systems so that the components of QEI provide genuinely secure, manageable, usable, and private space inside those systems. An even better solution for the long term will be to exchange today's vulnerable and cranky old operating system foundation for a more reliable, secure, and manageable one, while keeping most of the familiar user interface and application programming interfaces.

If we were into long titles, we might have called this chapter "The Operating System that Understands Whether You Are Indoors or Outdoors."

An operating system can either contribute to or detract from Quiet Enjoyment.

For starters, an operating system must obviously be reliable in order to contribute to Quiet Enjoyment. Apple Computer illustrates this point. After Apple replaced its old operating system with OS X – that is, it replaced the kernel with its NextStep subsidiary's Darwin variant of BSD Unix – the Apple Macintosh suddenly became the most reliable personal computer on the market. The Macintosh now tends not to crash. It just runs. BSD does its job without complaining. And remarkably, it does that without intruding upon the established Mac user interface.

What exactly makes the various Unix and Unix-like operating systems so solid and reliable? For one thing they don't use a rickety registry system, or a DCOM-style application component foundation. And they use sound memory management methods.

Stepping back, however, we might ask why they were built differently in the first place. Instead of simply noting better memory management we might ask what ac-counts for the fact that memory management was dealt with in a superior way when architectural decisions were made.

An obvious answer presents itself. Unix was designed with the assumption that many users would be using the computer at the same time. A famous remark on the subject that has been attributed to Microsoft's original Chief Software Architect shows a difference in design philosophy: "One's computer should be as personal as one's underwear." (Let's see…one computer for each member of your family, each with a separate operating system license and office suite license… we see where this is going…no wonder he's been the world's richest person.)

Unix got its start when computers were too expensive not to share. When the assumption is that many users, competent and incompetent, benign and malicious, using simple applications and complex ones, will be trying to do things all at once with this operating system, you tend to put robustness ahead of fancy features. Today the difference shows. Unix and its cousins are defensive. They handle all sorts of difficult and unanticipated situations gracefully, and keep on going in the face of adversity. Before we talk about features, isn't that where we want to start when we judge an operating system?

After the robust kernel was developed, the fancy features started to appear. Today many Unix-like systems deliver the same bells and whistles as Windows.

When it comes to technology products, we tend to equate quality with currency — the best are the ones that incorporate the latest technology. But in this respect, operating systems are different from other technology products. They're more like race horses, whose capabilities have a lot to do with their heritage. The latest technology is important, especially when it comes to things like driver support, but that stuff is easily added, as a trainer can prepare a good horse for a new course. Changing a horse's or an operating system's DNA to get better performance is a lot more difficult. Apple didn't try to make modifications to its old operating system, it simply replaced the existing Mac OS foundation with a version of Unix, keeping as much of the user and programming interface as possible. Apple replaced the new with the old.

Unix, Son of Sputnik

Unix actually got its inspiration from an operating system called Multics, which was a joint project of MIT, Bell Labs, and General Electric. Multics itself came from MIT's Project Mac and CTSS, which in turn were direct progeny of the one project that is responsible for a huge portion of America's technical competence.

That project was Sputnik.

Sputnik was of course not an American project, so let me explain.

My father was chairman of our town's school committee on October 4, 1957, when news of the launch of the Soviet Union's Sputnik satellite was splashed across the pages of the Boston Traveler. That night he convened an emergency meeting of the committee that adjourned at three in the morning, having produced a set of plans for immediately pouring effort and resources into improving the town's schools. That sort of thing was repeated in cities and towns all around the United States, making Sputnik, more than any other project, responsible for a rapid improvement in America's schools even while they were dealing with a huge baby boom.

Project Mac's goal was to make computing resources widely available to remote users with terminals. Ready access to those resources, the theory went, would help the U.S. recapture the lead in space and regain its self-respect as the world's technology leader.

The significance of that history is this: since Project Mac was launched in 1963, the world's best software architects, computer scientists, programmers, standards diplo-

mats, and users have been hammering on a small set of related code bases, always with the knowledge that the system is to be used by many users with diverse credentials, skills, and intentions.

Multics, MULTiplexed Information and Computing Service, was way ahead of its time. First presented as a design idea in 1965, made available in 1969, and with suitable performance and reliability coming a few years later, it was able to run on a symmetric multiprocessor; offered a hierarchical file system with access control on individual files; mapped files into a paged, segmented virtual memory; was written in a high-level language (PL/I); and provided dynamic inter-procedure linkage and memory (file) sharing as the default mode of operation. Multics was the only general-purpose system to be awarded a B2 security rating by the NSA.

One of the Multics developers from Bell Labs was Ken Thompson. That connection and others explain the resemblance between Multics and its nephew, Unix.

Another operating system (niece?) developed by a partnership of the same organizations — MIT and General Electric — was GCOS. The GCOS family tree is much sparser, showing few descendants. But GCOS had its own noteworthy design innovations, many of which are directly relevant today when we talk about close control of the authorizations of programs and the need for persistence as we tightly couple processes implementing office facilities over the Internet. Those ideas were kept alive in KeyKOS, a "persistent, pure capability operating system," and its own descendant, EROS (the Extremely Reliable Operating System) that runs on today's Pentium processors.

All of these operating systems started with the premise that many people would simultaneously use the computer on which they were running.

Back to the Future

Surprise! With the Internet, everybody is using your computer! If your computer is as personal as your underwear, then you have a lot of visitors in there. Better do something about it. And that something had better be more than a firewall.

Your view of personal computing may include computers as office equipment used by your employees. As we have pointed out, you don't have to take the plunge into telecommuting, with its open-ended tunnels, to expose your corporate information to outsiders. Wireless access points are proliferating like mushrooms in your network right now, provided innocently by workers who just want easy access as they move around and who have no idea of the vulnerabilities they are opening as they sneak a wireless access point under their desk.

Full-blown Unix is not really a personal computer operating system. But BSD and GNU (the progenitor of the Linux kernel) both offer a sound adaptation of the Unix approach, configured for personal computers and for tablets and phones. While Android is the best known Linux derivative for wallet-size devices, there are many others. There are even wristwatches that run Linux!

If we treat our Unix/BSD/Linux information appliances as disconnected islands, or as old-fashioned clients on a friendly, secure, isolated client-server network, then they are no better than more common personal computer operating systems. Let's face it, your PC and tablet and phone are now on the highway. They're right there on the median, or the break down lane, or the rest area parking lot — wide open, outdoors, there for the whole world.

Something must be done. Here's what Gartner has to say 77:

> Mobility is not an add-on to existing architectures: It is a profound disruption. Enterprises that ignore the impact of mobility on their software architectures are setting themselves up for accelerating software maintenance costs.
>
> In all the generations of IT architecture to date, there has been one constant – the entities that are linked by the architectures are computers. Mobility creates new rules – now the architecture must link the users of the systems. Of course, we always talked about users, but in practice the computers stood as proxies for the users. In a world where I have multiple devices, with very different capabilities, and I wish to move seamlessly from performing functions on one device to performing functions on another, how do the systems react? By logging me off one machine because I have logged on somewhere else (such as my cellular phone)? By breaking my connection and freezing my applications because I have moved between a wireless LAN and a cellular service? By sending me a 1,024x768 display complete with scroll bars so I can view all of it on my handheld device 320x200 pixels at a time?
>
> Mobility and device diversity require a new layer in the architecture. Worst of all, the "enterprise architecture" now becomes hopelessly "polluted," since it spans enterprise systems, carriers and devices that the enterprise does not control.
>
> Mobile device support is just one indicator of the emerging challenge of new device types (fixed devices embedded in control systems ranging from domestic to industrial, Internet appliances employing non-PC designs, and even variability in the designs of the PC). It is not feasible to accommodate this variety by assuming either that devices can mask the variability and call standard server interfaces, or that server-side systems can deliver output targets for each device type. The management of the interaction becomes an identifiable element alongside client and server at the highest level of system architecture."

Let's look at JetBlue, gaining profits and market share in, of all things, the airline industry. As every manager knows, employee morale comes from having everyone concentrat-

77 "The Impact of Mobility on Enterprise Architectures," from *Be Connected*, a Gartner/PC Connection newsletter, Volume 9, Issue 3 (September 17, 2002). Quote is from Gartner's *Strategy & Tactics/Trends & Direction*; Note: COM-16-7718, July 2002, S. Hayward.

ed in physical facilities so they can get to know each other in the context of the company culture. But every single one of Jet Blue's 600 reservations agents telecommutes from home.

So where does Jet Blue's great company culture come from? The answer is that workers at home can be less isolated, closer in fact to the organization and its culture in an online environment than in a forest of cubicles. Workers don't need to smell each other to get to know each other.

There is one thing, however, that makes the Jet Blue reservations office easier to implement than most offices. Information being shared in its online reservations office is more or less the same information that is available on Jet Blue's public reservations website — flight schedules, fares, available seats, etc. – not exactly sensitive inside information.

We can safely assume that Jet Blue's planned disclosures for the upcoming quarterly conference call with securities analysts are discussed elsewhere.

That does not mean that such sensitive information is not shared online. Top management perhaps more than any other part of the organization, needs to share spreadsheets and databases and projections right now, not when the CFO gets back from a trip to Sydney, at which time the VP of Sales will be in Vienna anyway. The days when sharing of important files could wait until everyone was in the same physical room are over. If you are not sharing important information at all levels online, and if the online facility cannot be flexibly expanded to include rooms where suppliers, distributors, partners and important customers can meet and share files, then you have a problem.

Is your company prepared for a future where this is the norm? Is your competitor? One way to get prepared is to adopt an operating system where telecommuters, including those that handle confidential information, can work indoors, where that information can be protected. Remember, there is no such thing as a "subnetwork" with such niceties as network address translation (NAT) to protect your information when your workers telecommute. Rather, there may be NAT but it's their own network addresses that are being translated, not yours. The end of your tunnel is outdoors.

Come InDoors

Dorren™ is a new operating environment that will allow you to build online real estate facilities with ease, and at the same time will make your computer more useful. Dorren "understands" the difference between InDoors and outdoors. It knows whether you have authenticated yourself and whether you are in an authenticated online environment. It enhances the security and utility of the system for you.

The core of Dorren is not new at all; it's a combination of elements that "know" what is really required of building codes: Osmium-compliant PEN Component, identity from Identity Reliability Component, and an occupancy permit that is issued in compliance with the Building Codes Component.

To make use of InDoor facilities you'll need to enroll and obtain a Foundational Certificate.

But for the time being, you can come InDoors without an Osmium-compliant operating system such as Dorren. You can make use of the other 11 components of the Quiet Enjoyment Infrastructure without an InDoor operating system. Your employees can use any browser that understands digital certificates, operating on any computer with any operating system to get to a code-compliant online facility.

It will be compliant with current building codes, simply because we have to start with what people already have. We call it the Montaigne principle, living with the living, a necessary but uncomfortable compromise with dismal present realities. Over time the codes will become more stringent, and at some point will require both hardware and operating system compliance with the Osmium standards.

Why an Open Source Kernel for Dorren?

Reality can be counter-intuitive. Those who are familiar with open source and proprietary software know that open source products are often more secure than proprietary ones. Bruce Schneier explains why[78], using data from the Report of the Director of the Administrative Office of the United States Courts on Applications for Orders Authorizing or Approving the Interception of Wire, Oral, or Electronic Communications to illustrate. The report notes that

1. Encryption of phone communications is very uncommon. Sixteen cases of encryption out of 1,358 wiretaps is a little more than one percent. Almost no suspected criminals use voice encryption.
2. Encryption of phone conversations isn't very effective. Every time law enforcement encountered encryption, they were able to bypass it. I assume that local law enforcement agencies don't have the means to brute-force DES keys (for example). My guess is that the voice encryption was relatively easy to bypass.

These two points can be easily explained by the fact that telephones are closed devices. Users can't download software onto them like they can on computers. No one can write a free encryption program for phones. Even software manufacturers will find it more expensive to sell an added feature for a phone system than for a computer system.

This means that telephone security is a narrow field. Encrypted phones are expensive. Encrypted phones are designed and manufactured by companies who believe in secrecy. Telephone encryption is closed from scrutiny; the software is not subject to peer review. It should come as no surprise that the result is a poor selection of expensive lousy telephone security products.

78 "Encryption and Wiretapping," by Bruce Schneier, *Crypto-Gram*, May 15, 2003.

> For decades, the debate about whether openness helps or hurts security has continued. It's obvious to us security people that secrecy hurts security, but it's so counterintuitive to the general population that we continually have to defend our position. This wiretapping report provides hard evidence that a closed security design methodology — the "trust us because we know these things" way of building security products — doesn't work. The U.S. government hasn't encountered a telephone encryption product that they couldn't easily break.

And then there is this, from The Register[79]:

> Thomas Reed's At The Abyss recounts how the United States exported control software that included a Trojan Horse, and used the software to detonate the Trans-Siberian gas pipeline in 1982. The Trojan ran a test on the pipeline that doubled the usual pressure, causing the explosion. Reed was Reagan's special assistant for National Security Policy at the time; he had also served as Secretary of the Air Force from 1966 to 1977 and was a former nuclear physicist at the Lawrence Livermore laboratory in California. The software subterfuge was so secret that Reed didn't know about it until he began researching the book, 20 years later... Soviet agents had been so keen to acquire US technology, they didn't question its provenance. "[the CIA] helped the Russians with their shopping. Every piece of software would have an added ingredient," said Reed to NPR's Terry Gross last week...
>
> Tools you can trust
>
> ...Closed source software vendors such as Oracle and Microsoft hardly need to be reminded of the delicacy of the subject. A year ago China signed up for Microsoft's Government Security Program, which gives it what Redmond describes as "controlled access" to Windows source code. But the Windows source itself doesn't guarantee that versions of Windows will be free of Trojans. Governments need access to the toolchain - to the compilers and linkers used to generate the code - as that's where Trojans can be introduced. Without tools source, licensees are faced with the prospect of tracing billions of possible execution paths, a near impossible task.
>
> Until the closed source vendors open up the toolchain, and use that toolchain for verifiable builds, this is one area where software libre will have a lasting advantage.

Relying upon software whose source code is not available for scrutiny is a risky thing to do. The rash of spyware that people have been experiencing should be enough to

79 "Explosive Cold War Trojan Has Lessons for Open Source Exporters," by Andrew Orlowski, *The Register*, March 16, 2004.

convince anyone of that. Someone has planted code in your computer or phone and it could be sending anything to its masters — you'll never know what. Software that is to be relied upon must come from open sources.

Dorren's Indoor Space

The first step in the path to widespread availability of Osmium-certified personal computers and other information appliances will be the production of a universally-installable secure virtual machine (VM), which we call InDoors™. As the name implies, InDoors hosts the real estate — the collaborative facilities we have been discussing.

It is not a space that accommodates anyone who wants to do software development work. Only code that is personally signed by an individual who is certified under the standards of the Professional Licensing Component can operate within the virtual machine, just as only certified architects and structural engineers can design physical buildings. An identifiable individual needs to be professionally accountable for any problems. Effectively InDoors makes the computer disappear.

A number of virtual machine technologies were considered as the basis for InDoors. The most intriguing VM is the operating system Inferno from Vita Nuova Holdings Limited. Inferno is a derivative of Bell Labs' Plan 9, which is an operating system designed by many of the original developers of Unix. Inferno installs either as a standalone operating system or as an application under any of a number of other operating systems, including most versions of Windows.

Our real estate model tries to ignore physical and network-layer barriers, such as routers and firewalls, and instead defines boundaries through the use of facilities definition files. In other words, if two computers are tightly coupled in a computer room in Singapore, a computer in Denmark might be "closer" to one of them than the second processor in Singapore is, if that is how the facilities description files define the facility. The design of Inferno inherently supports such a structure.

Indeed, this particular virtual machine natively does what others are struggling to find a way to do. "Grid computing" serves as the budget-building rallying cry among vendors and customers attempting to retrofit existing systems to do what the inventors of Inferno anticipated in its original design. Remember, that includes people who, having invented Multics and Unix decades ago and having stayed decades ahead of the pack, realized that the old expression "the network is the computer" would someday really mean something. That someday is today.

While Inferno was actually forked from Plan 9 by Bell Labs, its development as a robust operating system + virtual machine is largely the work of Charles Forsyth and his team at Vita Nuova in York, England. Inferno and Plan 9 have themselves spawned interesting indoorsy derivatives, most notably from a team led by Francisco Ballesteros at Universidad Rey Juan Carlos in Madrid.

Dorren will borrow from these brilliant open source developments, by involving operating system and virtual machine developers who "grok" the advantages of building, managing and using information facilities in the manner used by architects and contractors and building inspectors and occupants in the world of physical buildings.

With Dorren, processors around a network are like cells in a body. Indoor spaces are defined in "blueprints," or small programs written in programming language such as Limbo or Go. Dorren's virtual machine manages the indoor spaces and is ultimately responsible for the security of the facilities that it hosts or in which it participates.

The Walled Garden

The most secure building in the world should be located in a town or office park guarded by an active security force. The perimeter of the yard of a residence should be considered a real boundary and should be supported by substantial security measures, even though it is outdoors. We need Osmium compliance for the whole environment, including the outdoor space that is within the perimeter of the yard or community where we live or work.

The outdoor space, the walled garden, constitutes the operating system environment where we run the applications that we depend upon, in spite of their vulnerabilities. We can't suspend our use of word processing, spreadsheets, presentation programs, databases, while they are rewritten to work indoors. Unfortunately we will need to do our solitary work at a park bench for awhile. For the next few years anyway we will need to create our files outdoors and share them indoors.

At least we can put the park bench in a walled garden, that is, a traditional operating system that operates on top of Dorren's virtual machine. Some candidates are OpenBSD and Linux.

The Windows emulator called Wine and its commercial cousin Crossover Office enable you to run Microsoft Office on Linux. A number of software products implement the Windows application programming interface and other elements of the Windows environment to make that possible.

Better yet, the free Libre Office office suite provides almost everything that Microsoft Office provides, and its code is open for inspection, audit, and signing by an individual code auditor, professionally licensed by the City of Osmio Professional Licensing Department.

Integrated Cryptography

Any operating system that can properly use key pairs in an authentication process, and establish a session key to carry on after authentication, should be usable to access a QEI-based facility today. On the other hand, as we have noted, what good is the process if the operating system itself is vulnerable?

The walled garden portion of Dorren should be as well protected as possible. It

should work tightly with cryptographic hardware that is being introduced into computers and should implement the kind of crypto-integrity that is needed to gain the full benefit of the PEN Component. When a secure operating system kernel and the PEN Component are tightly integrated, we are talking about Osmium compliance.

As usual, Moore's law and good software conspire to make old impossibilities possible. Today sound cryptographic processes can be built into everything. The lights no longer dim when you try to use RSA or AES in desktop and laptop computers, and efficient elliptic curve cryptography has at last found acceptance in phones, tablets and other ARM-based devices. The PEN Component of QEI assures us that pervasive digital signatures from reliable identities and pervasive encryption are readily available to both InDoor spaces and walled garden spaces.

Popularizing Dorren

Open source operating systems and applications have gained significant share of installations on servers and embedded clients, but negligible penetration on desktops. The most significant obstacle has been the fact that open source committer teams and their broader communities are quite busy with development work and don't have the resources to appeal to and support new users. Often they get irritated by questions from new users who have not taken the trouble to read the documentation. Being told to go away and read the manual does not win friends for open source.

Of course there is a major difference in the set of assumptions. At one extreme is the community of OpenBSD committers and users, members of which can be quite blunt about the fact that they don't care whether they attract new users or not. At the other end is the user who is used to being treated as a customer and has not digested the fact that the product was provided by volunteers.

While the committers ought not be distracted by making a new, popular version, it makes no sense to have literally hundreds of millions of computer users putting up with insecure, unstable, buggy, manipulative, hidden-agenda-laden operating systems when a better alternative is available.

Who's going change that? Who's going to hold the hands of people who are accustomed to Windows because that is what came with the computer they purchased at Best Buy, and who rely upon others for anything related to configuration and networking? What kind of organization is built upon the premise that the user is the most important person in the world?

Of course that is practically the definition of a successful, customer-focused commercial enterprise. Commercial enterprises provide service to new users while distracting the committers only long enough to notify them of money being sent, and perhaps to make an occasional request for a special feature in the kernel.

Being able to do whatever you want with a piece of software, and giving back to the community of developers when you add code to their work, is what open source is all

about. Building a commercial product from open source components, a la Red Hat or Apple, falls within the being-able-to-do-what-you-want part. At the same time, the process must have integrity.

Tim O'Reilly once noted that "anyone who puts a small gloss on a fundamental technology, calls it proprietary, and then tries to keep others from building on it, is a thief." That's an extreme of the sort of thing we want to avoid. A commercial enterprise should add value to open source by researching what customers are likely to want and need, investing in the integration of components, building an audience education effort (e.g. by publishing books for wide audiences), marketing and brand-building, providing a means by which customers can get service and support, and helping open source developer groups with user support services, feedback, and a portion of the product's earnings.

Putting It All Together

Dorren will support all the QEI Components and will serve as a complete software platform for most users. It is a client system that can easily be transformed into a server if the machine has the proper capacity and configuration and its user knows how to run a server. A facility (building, office suite, residence, fraternal meeting hall) can be "served" from a computer that is not configured as a server; however, for the facility to be readily available to its occupants it will be best to operate it from a bona fide server. Most importantly, Dorren knows whether at any moment those occupants are indoors or outdoors, and behaves accordingly.

Dorren does not know the difference between a browser and a desktop. The U.S. Justice Department fretted over Microsoft's integration of the browser into Windows. But everyone else knows that integration is what users desperately need. Users crave a client package that knows how every component works, where surprises are made nearly impossible, where plug and play is reality. But everyone worries about the power that such a desktop gives to the company that controls it.

Your Dorren machine will be under your control. Dorren is not only secure and robust and reliable, it is honest. It has no hidden agendas lurking inside hidden pieces of code. If you use your copy of Dorren to host a collaborative environment — that is, a building — then in order for it to be certified as secure the building must have an occupancy permit. The process by which that is obtained is open and public and designed to resemble as closely as possible the process by which an occupancy permit is obtained for a physical building.

You Can Fill the Power Vacuum

People love to debate whether Windows will always rule the desktop. The demise of the platform that totally dominates a space is a repeated theme in information technology. It is not a design issue, not a religious issue; it has nothing to do with whether or not

you like the incumbent. It just happens, in the same way an epidemic subsides for no apparent reason. IBM experienced it; DEC experienced it. Next it will be Microsoft's turn. We've all heard that nature abhors a vacuum. What will replace Windows?

Dorren will.

When will all this be reality? Sooner than open source development schedules would seem to indicate. The pace of open source development suffers tremendously from its economics. Too often, a brilliant piece of software gets stuck at version 0.93 because its key developers need to shift their focus to something that will more immediately put food on the table.

Dwell for a moment on the similarities between the process of designing and building software and designing and building buildings. Then dwell some more on the magic ingredient in the latter that ensures that the professionals involved in it get paid for their hard work.

In the next chapter we'll show how the Professional Licensing Component adds sound economics to the process of open source development, so those who put heart and soul into the development and deployment of Dorren can be well compensated.

When will Dorren be a commonplace reality on information appliances around the world? Perhaps it's up to you. With superior design and code heritage, the best talent, and new developer economics as a starting point, join us as we go forward and ensure that the best horse is ready to win this race.

To see the current state of development of

The InDoor Operating System

…and to learn how your

experience with distributed OS protocols such as P9P

might be put to use in its development, please go to the InDoor Operating System Development Office at osmio.ch