

What Exploring Space Taught Me About Systems Architecture and Personal Computing

Patrick F. Wilbur

Department of Computer Science,
Clarkson University

November 28, 2011

Copyright Notice

Copyright 2011, Patrick F. Wilbur.
Last modified: November 28, 2011 8:06 PM EST.

Some images have been taken from Public Domain or other copyrighted sources. For more information concerning the licensing of those images, please consult the References list at the end of this document. The author of this work believes that use of those images constitutes Fair Use according to U.S. Copyright Law.

==

LICENSE:

Patrick F. Wilbur
Department of Computer Science
Clarkson University
Potsdam, NY USA

<http://pdub.net>

These slides and content are released under the Creative Commons Attribution-Share Alike 3.0 Unported license, available online at <http://creativecommons.org/licenses/by-sa/3.0/>

You may share (copy, distribute, and transmit) this work, and remix (adapt) this work, as long as you attribute this work to the author and share adapted works under the same or similar license by leaving this entire notice in place (including the original author's name/contact information/URL and this license notice).

Acknowledgments

Special thanks to my advisor, Dr. Jeanna Matthews, for her virtualization, virtual appliance, and virtual machine contracts support, interests, and insights.

Special thanks to my friend, Dr. Todd Deshane, for helping with OSCKAR Core implementation, asking me difficult questions, and using OSCKAR Core in his work.

Research Topic

- Intersection of:
 - **Personal computing**
 - **Security**
 - **Usability**
- Virtualization as a means to contribute to these areas by:
 - Improving **software distribution**
 - Improving **data protection**
 - Maintaining **trust** (established security & reliability)

For my research, I am looking at the intersection of personal computing, security, and usability.

In particular, I'm looking at using virtualization (as well as hardware-assisted virtualization) as a means to contribute to these areas by improving software distribution, improving data protection, and, most importantly, maintaining trust (as it relates to security and reliability of both data and the system).

I mean trust, as in trusted system. A trusted system is defined as a system that is relied upon to enforce security and reliability policies.

Let's begin with a real-world illustration.



In April, several of us, in conjunction with the K2CC Amateur Radio Club here at Clarkson, launched a high-altitude weather balloon to the edge of space...







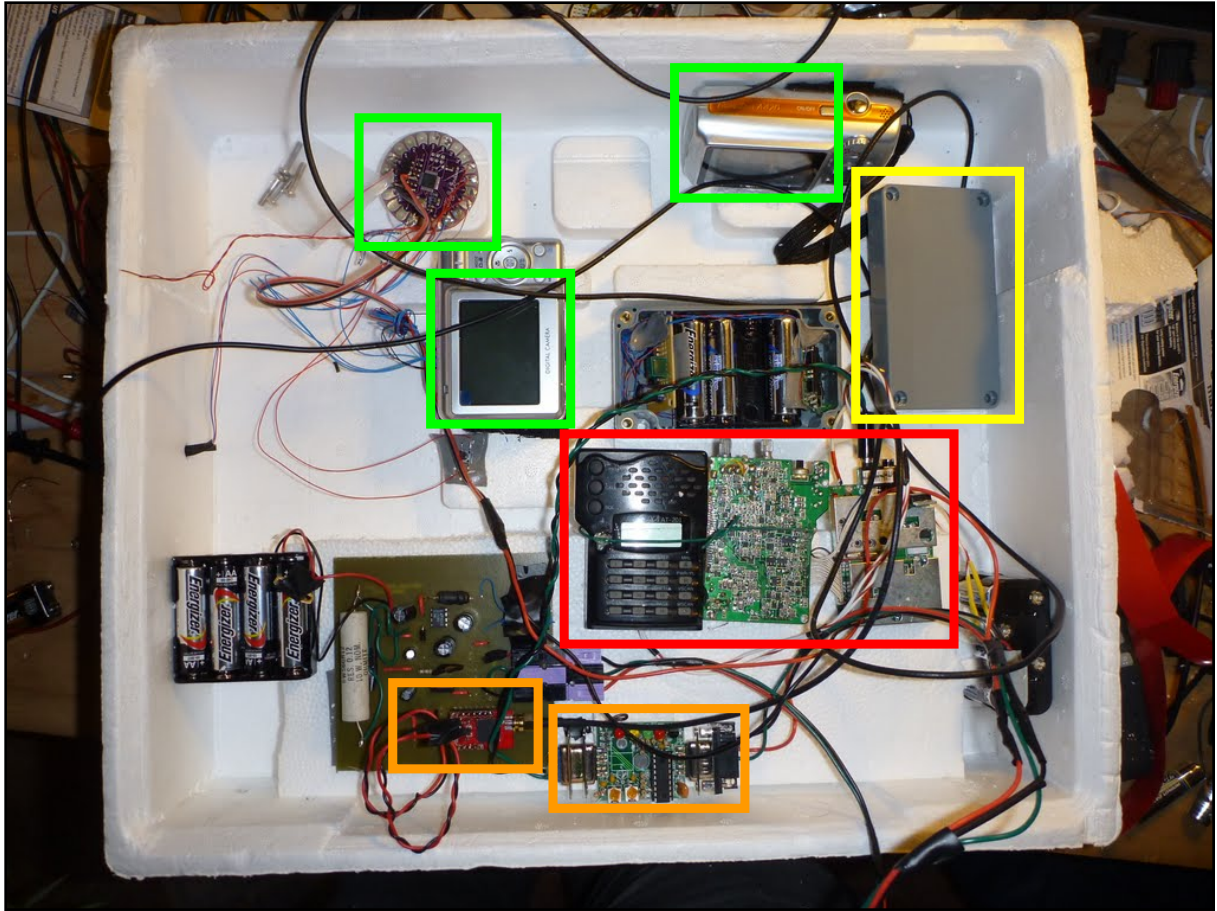


So, what does our space expedition have in common with improving the personal computing experience?

While an embedded general-purpose computer with various peripherals could have easily been programmed to perform our task, for ensuring reliability we instead ...



... separated the space capsule's functionality into several single-purpose subsystems, each with a small hardware and code base.



Systems indicated by orange were in charge of gathering GPS data and modulating into an analog waveform using a modem,

systems indicated by red were in charge of transmitting telemetry waveforms,

systems indicated by green were in charge of photography and automatically operating cameras,

and systems indicated by yellow were in charge of emergency beacon functionality for locating the space capsule by foxhunting in the forest

So, while there is a single overall problem of performing a space expedition, we were able to divide the problem into several requirements, and each subsystem was designed to address a requirement as a part of our overall solution.



The stability of our space capsule system depended upon reducing the size of the hardware and software base that we needed to test and to trust for each task

Some Computer Security Principles

- Reduce size of **trusted computing base (TCB)**
- Apply the **Principle of Least Privilege (POLP)**
- Attempt to **understand user intent**
- Attempt to enforce **isolation** between applications

1. [Talk about bullet points.]

2. Can anyone define TCB?

Definition of TCB [second is better]:

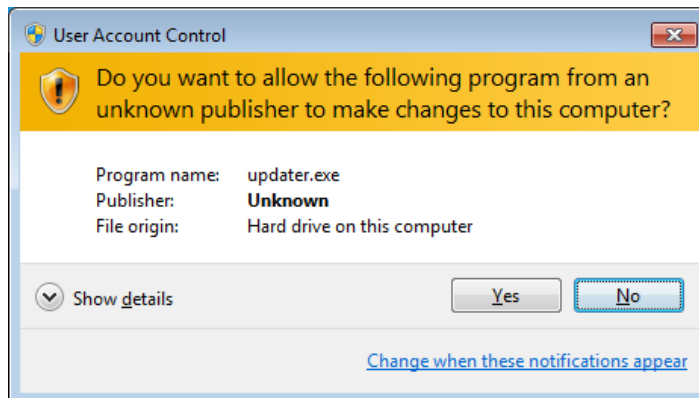
" In the classic paper, 'Authentication in Distributed Systems: Theory and Practice'[2], Lampson, et al., define the TCB of a computer system as simply:

'a small amount of software and hardware that security depends on and that we distinguish from a much larger amount that can misbehave without affecting security'."

"The Orange Book, another classic computer security literature reference, says that the TCB is 'the totality of protection mechanisms within it, including hardware, firmware, and software, the combination of which is responsible for enforcing a computer security policy'."

Classic Application of Principles

- Operating systems already do attempt to reduce TCB
- **Mandatory Access Control** (e.g. SELinux) for POLP
- **User Account Control** (UAC) for user intent
- Qubes OS (VM **sandboxing**) for selective isolation



[List these things that have been done.]

There are limitations though:

1. Operating systems, while considered trusted one minute, can be entirely reconfigured the next due to software updates or malicious activities
2. MAC systems like SELinux are complicated to use and are usually disabled [anything else to discuss here?]
3. For this talk, let's skip over the user intent issue and current limitations, although I have been studying that
4. Isolating hand-selected applications for the purpose of testing doesn't offer the simplicity a desktop PC solution needs

Idealistic OS-app Model

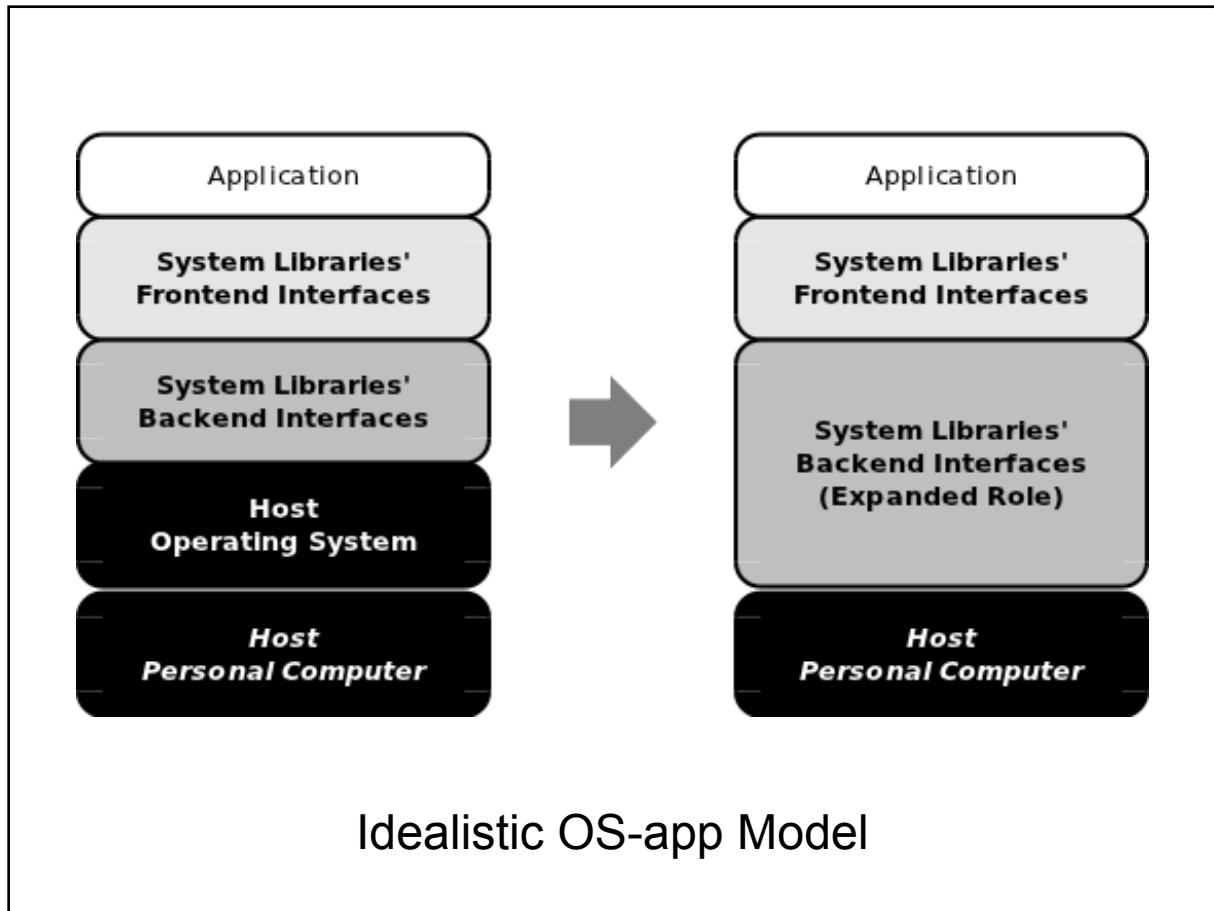
- Historically, hardware provided multitasking/multi-tenancy
- **OS-apps:**
 - Talk to hardware without needing (much of) an OS
 - Depend upon a virtualization hypervisor for multitasking
- **Desktop Decomposition:**
 - Reduces size of TCB
 - Enforces extreme isolation
 - Isolation helps make POLP enforcement easier
 - Few-purpose systems easy to watch, secure



On IBM's System 370, platform virtualization was introduced, actually, as a solution for the problems of multitasking and multi-tenancy

[Define OS-app]

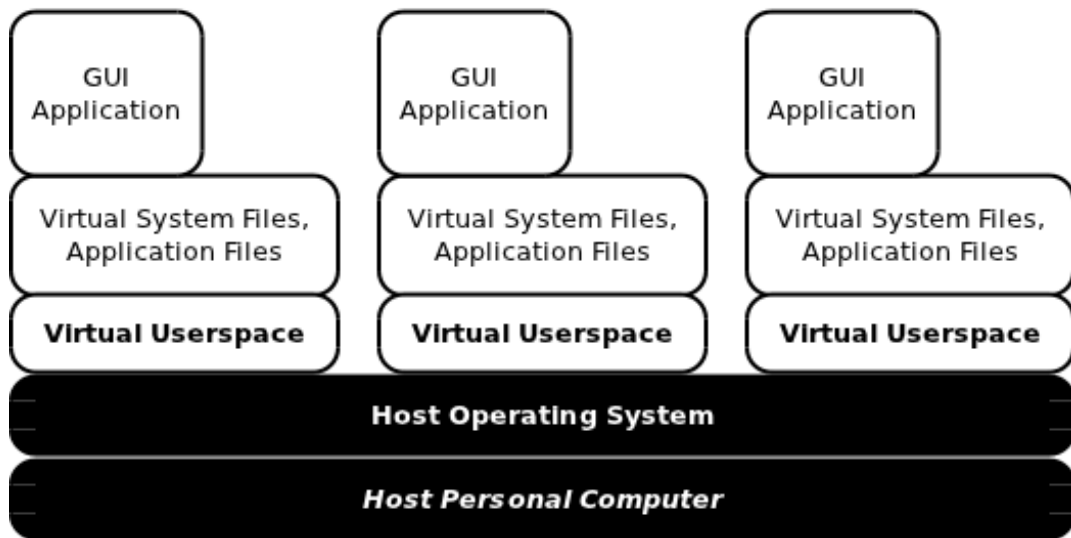
Desktop Decomposition is essentially the equivalent to what we did in our space capsule--taking what would otherwise be a general-purpose computer and dividing up tasks into simpler, fewer-purpose components. This is great, because the general-purpose computer is a little too reprogrammable most of the time, since we don't want it to be able to become insecure or unstable. [Discuss bullet points]



[Describe picture]

Software designed to run on microcontrollers, such as what powered our homemade space capsule, already functions like an OS-app, pictured on the right

Due to the dynamic linking of libraries, building virtual OS-apps on PCs should, in fact, be possible without necessarily needing to redesign or even recompile existing software; however, ...



Desktop Decomposition using Virtual Appliances

...it is still easier to simply use the idea of virtual appliances--or applications contained within virtual machines running entire operating systems--to perform desktop decomposition

[Describe the picture]

Benefits of Desktop Virtual Appliances

- Reverses disturbing trends:
 - SaaS/AaaS (Software as a Service, Application)
 - Software requiring installation/execution as Administrator
- Hardware more predictable than software-defined systems
- Reduces size of TCB: easier to engineer trustworthy (secure, reliable) systems
- Rollback (on demand, on application close, etc.)

2. While hardware can still contain security and stability vulnerabilities, it is easier to prove a limited number of possible states that a hardware system can enter than it is for a software system. For instance, since a software system can be reconfigured upon intrusion, memory corruption, or software update, a software system can be completely different than it was the day before. Apart from re-imaging an integrated circuit chip or re-etching a circuit board, hardware doesn't have the ability to drastically change from one minute to the next (except for failures). One can attempt to include aboard their mission-critical weather balloon a general-purpose computer, software defined, with programs stored in RAM, but I highly recommend against that.

3 & 4. [Important to note] Software that requires administrative access for installation or execution actually becomes a part of the TCB! Under a virtual appliance model of personal computing, this can be entirely avoided, preserving the integrity of a smaller, underlying TCB.

Virtual Machine Contracts

- Define expected *operating environment*
- Optionally define *expected operating characteristics*:
 - IDS
 - Resource usage patterns
- Allows distributing virtual appliance disk image or not

Virtual machine contracts define the expected operating environment for a virtual appliance, and, optionally, the expected operating characteristics (which can help identify problems with virtual machines). [Eli Dow's research, for instance]

Virtual Machine Contracts

- **OSCKAR Core:**

- VMC processing/enforcement engine
- Provides an automation platform for developing virtualization-related solutions
- Service provider/consumer architecture makes it extremely versatile and extensible

Virtual machine contracts define the expected operating environment for a virtual appliance, and, optionally, the expected operating characteristics (which can help identify problems with virtual machines). [Eli Dow's research, for instance]

Open Problems

- **Software licensing**
- Determining **user intent**
- **Scalability** (mem. deduplication? other types of virt.?)
- **Hardware vulnerabilities** (security issue, not a trust issue)

Current Progress & Future Work

- Build OSCKAR Core (for automation)
- Build/test Appify tool (first generation, with poor delegation)
- Implement basic DataStore (provides selective sharing)
- **Test power consumption overhead**
- **Additional performance testing** (some done before)
- **Improve Appify tool** (with preemptive worker VM creation)
- Implement user intent systems
- Incorporate user intent into improved User DataStore

Questions?



No, that is not a pinata--it's the space capsule

Any questions?

References

- URL: http://en.wikipedia.org/wiki/Trusted_system
- URL: http://en.wikipedia.org/wiki/Trusted_computing_base
- URL: http://en.wikipedia.org/wiki/System_360
- URL: [http://en.wikipedia.org/wiki/VM_\(operating_system\)](http://en.wikipedia.org/wiki/VM_(operating_system))
- URL: <http://pdub.net/projects/near-space-weather-balloon/>
- URL: <https://picasaweb.google.com/100543486411711719984/TheGreatGigInTheSky>
- Paper: Joshua Sunshine, Serge Egelman, Hazim Almuhammedi, Neha Atri, and Lorrie Faith Cranor. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. Proceedings of the 18th conference on USENIX security symposium (SSYM'09). 2009. USENIX Association, Berkeley, CA, USA, 399-416.
- Paper: Wilbur, Patrick F. and Todd Deshane. Johnny Can Drag and Drop: Determining User Intent Through Traditional Interactions to Improve Desktop Security. CHiMiT '10: Proceedings of the 4th Symposium on Computer Human Interaction for the Management of Information Technology. November 2010. DOI: 10.1145/1873561.1873565