Date	Commentor	Section	Comment	Comment Status	Updated By
1/30/18	Judson Neer - EveryoneCounts	14.2-A	what the requirement is getting at, a strict reading of the requirement could lead to prohibition of interpreted languages such as Python or Perl, which seems overly-restrictive. It would also preclude the use of the Android Run-Time (ART) since it's a just-in-time compiler, and probably a number of other technologies I'm not thinking of. It seems better to me to let the requirement statement stand without an attempt at enumerating examples. Of course this leads to another problem, which is the definition of extraneous. We should seek to define that term more carefully. I'm interested in what the larger group has to say, but perhaps something along the lines of this: The voting system must prevent processes and services that are not required for system operation from	I definitely understand the concern here. Resolved by moving the examples to the discussion section.	Gema
1/30/18	John Dziurlaj	14.1-A	ISO Standard withdrawn	Updated to correct ISO	Gema
1/30/18	John Dziurlaj	14.1-A.1	Where do these documented threats come from?	The threats are identified during the risk assessment process. Perhaps the group can identify baseline threats that need to be included	NIST / MITRE group
1/30/18	John Dziurlaj	14.2-A	Agree with Neer's Comment	Compilers, can be modified to allow run-time compilation.	Marc Schneider
				Any service not needed for the system to function. E.g. if the system doesn't use a printer, then print spooler services should be disabled by default. / Updated title	Marc / Josh
1/30/18	John Dziurlaj	14.2-B	What are these? All services?	I agree that "other services" seems vague. Can we add something a little more descriptive like "extraneous" or "non-essential"? Added non-essential	Gema
1/30/18	John Dziurlaj	14.2-C	I would prefer a physical indicator (e.g. a LED) that is controlled by networking firmware. An attacker could potentially change the voting system code to disguise the true status of a port.	Agree on the physical indicator, but networking firmware can be modified as well. / The indicator itself is not meant to stand-up against attacks on the voting system. We have other controls such as software whitelisting and malware detection that help to prevent those types of attacks. This is meant to alleviate accidental misconfigurations.	Marc Schneider / Josh
	John Dziurlaj	14.2-D	See previous comment.		
1/30/18	John Dziurlaj	14.2-I	A misreading of this requirement may lead to less secure systems. For example, if library X has 100 functions, but the voting system only uses 10, then the manufacturer might choose to extract just those functions from that library. If a security venerability is discovered and a patch released (by the library's owner), the patch will not be easily applied, leaving the system vulnerable for longer.	The build process should be able to handles this scenario. / Discuss with working group, consider deletion.	Marc Schneider / Josh
				Tamper evident tape or seals covering physical ports would accomplish this. Maybe revise to "must be able to restrict access"	Marc Schneider

1/30/18	John Dziurlaj	14.2-J	I'm not sure how this is accomplished.	I agree with Marc. Because it is not something the voting system actively does, I have changed the requirement to include Marc's suggestion.	Gema	
1/31/18	Lauren Massa-Lochridge	14.2-K, 14.3	Malicious code can, for example, can be embedded in firmware or in devices that are components such as storage devices (or removable media), or other hardware components of a system. The vector is not necessarily mitigated when the component is 'cleared' or re-programmed or re-formatted (in the case of storage), as the case may be depending upon the nature of the hardware or firmware component of a system. A while back I posted pointers to standards used in the semiconductor industry for supply chain vulnerability prevention or mitigation.	I agree. I was not sure if we could make a requirement for supply chain or how it should be worded. I think my thoughts were that boot validation/software validation/whitelisting might solve this. I think supply chain is a well-known threat to voting systems. Maybe we should call it out. "The voting system must provide documentation for all components that are not directly manufactured by the voting system vendor." Or should we require documentation for SCRM? **I added a potential requirement into the document.	Gema	
2/1/18	John Sebes	To Judson	Judson, Thank you very much! And I agree with the principle that there is a goal of a VS component not being able to have new code plopped and run on it, while prohibition of compilers is down in the weeds of one of many possible methods of pursuing this goal. And as much as like your simple statement > /The voting system must prevent processes and services that are not > required for system operation from being installed or executed./ I think that the term "prevent" is problematic. I completely support the idea that a VS component should use any means practical to lock down its SW base. And I'd add that the test and certification process should be determining whether the VS maker's document claims of lock-down mechanisms are matched by testing.			

		continued	But at the same time, 100% is a terrifically hard goal. Just consider the idea of a perfect execution white-list reference monitor, and never mind that creating one requires a solution to the halting problem :-). If such a mechanism existed, it would depend on stored data of some sort. It would be dependent on the firmware of the data storage device. The firmware could be malicious, with targeted malware that returns modified versions of the WL data, enabling malicious software to run. Run can't expect VS vendors to solve that problem. Rather, they need a software tampering threat model (excluding HW threats), and testing to demonstrate that the system integrity enforcements meet the requirements of the threat model. AND we should expect vendors to comply with the best practices of supply chain risk management, in order to mitigate the risks of hardware level attacks that can undo system integrity enforcement. > I think that the term "prevent" is problematic.	The last statement here make me think we should include a requirement on SCRM.	Gema
2/1/18	Judson Neer - EveryoneCounts	To John Sebes	John, I see your point completely. As much as I otherwise dislike the un-testability of requirements containing hedge words like "minimize" or "to the greatest extent possible", there might not be an alternative here. A possibility: "The voting system must prevent, to the greatest extent possible, processes and services that are not required for system operation from being installed or executed." Or perhaps we can replace "must" with "shall" to allow for the practical limits of this requirement.	I'm starting to understand the concern as this discussion continues. It's very helpful for me to understand the potential restrictions this requirement may apply. But I'm still not sure "to the greatest extent possible" is the right approach. I definitely think we should bring this topic up on the call. I'm interested in hearing more feedback from the group.	Gema
2/5/18	John Sebes		Yes, I suppose that "shall" and "as possible" caveats are appropriate though not thrilling due to making a requirement not exactly required. Harking back to ITSEC and TCSEC days, the way that I'm familiar with these non-requirement requirements is this: the guidance document requires a system builder to include mechanisms for X (in this case preventing the install or execute of un-needed software) and requires them to document their mechanism's design, specification, and test plan. Then, it's up to a test lab to try to understand the mechanism, determine whether its function meets the functional requirement for X, to recreate the builder's test procedures and repeat the builder's test results, and to document the level of demonstrated effectiveness of the mechanism. And then up to an accreditor to review the test labs findings and determine whether the tested system meets the requirement X "sufficiently."		

		continued	I agree with you that there is a problem with specificity, but I think that that is a consequence of the ambitious scope of these standards -- to define security for system-of-systems (a number of interoperating distinct physical and logical components), without ever actually defining the individual systems. To overcome the lack of specificity, what may be needed is something like an ITSEC "target of evaluation" (TOE) definition for each discrete component of a voting system. A TOE that is standard and can be re-used across multiple voting system products would be parallel to the approach that has worked well for NIAP for security C&A of common product categories.			
2/7/18	Judson Neer - EveryoneCounts		Back in my days working for a military contractor, we hardened our systems using the Security Technical Implementation Guides (STIGs) published by the Defense Information Systems Agency. Perhaps there's some value in considering how they approach the issue of minimizing extra operating system software/services. STIGs are operating-system specific and checklist oriented, so not quite appropriate to use wholesale in the VVSG. Taking a quick look at the STIG for RHEL 7 (available for download at https://iase.disa.mil/stigs/os/unix-linux/Pages/index.aspx), there isn't a comprehensive "minimize system services" requirement, but rather detailed steps for disabling known superfluous users, processes, etc. The nice thing about such a checklist is that it is concrete and verifiable. The downsides are that they're too narrowly-defined for general applicability across all possible operating system configurations. John, perhaps the best we can do is as you proposed, namely to require documentation and test steps	We have requirements about implementing secure configuration / hardening guides. This has been made more clear in 14.2-E	Josh	
2/8/18	John Sebes	STIGs	Good point about the STIG! I see top down and bottom up relevance. Top down, I can envision a version of the RHEL7 STIG that's tailored for embedded system devices that should not have network connectivity, and only run a fixed set of user-space software that doesn't require GUIs. That would be considerably more specific I believe that traditional ballot scanners and tabulation manager software should fit this description. Bottom, there is also a fine set of tools for RHEL that enable the definition of a system build that starts just with the kernel and user-space daemons for init and devices, and adds on only what the system builder needs. In this approach, a system builder should be able to list *every* single building block and document why it is needed. A worked example of the latter is here: https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FTrustTheVote-Project%2FDeviceManager&data=02%7C01%7Cgema.howell%40nist.gov%7Cf6053b2ed0e24c07ac5e08d56f594f7a%7C2046e5e15ac543e384f33cedc35a5eb2%7C1%7C0%7C636537351730064966&sdata=BCQtQdpqMH2skwyyQ9OJdmuaB6yYGBj2duojMbiz6rc%3D&reserved=0			
		Continued	We also have other efforts underway to use SE Linux and other kernel security modules to enforce controls on execution privilege, and a mandatory security policy limiting the privilege of user space code. That stuff might also fit into a STIG for an embedded system.			

2/9/18	Bernie Hirsch	SCAP	Are you familiar with SCAP? https://scap.nist.gov/	The SCAP program is relevant, would be interested in hear about how you see it fitting in.	