Response Letter

We really appreciate the comments and advices from reviewers. Some comments really help us a lot, we followed them and made the corresponding modifications to this manuscript.

Q1: Both of the two reviewers mentioned that, if HyperPS (our work) has token ROP attacks into consideration. The first reviewer mentioned that if the attacker could crash the HyperPS Space by using a dedicated kernel page table though CR3-targeted ROP attack. The second reviewer raised a similar question. He questioned whether an attacker could tamper VMCS and EPTs with ROP payload or not.
A1: For the ROP attack (code-reuse attack), we have taken it into consideration when we analyzed the threat model and designed HyperPS.

According to CVE statistics, most modern ROP attacks (Code-Reuse Attacks) try to invoke functions with illegal control flow or invoke the legal function with dedicated illegal parameters. In this paper, we hooked the most fundamental functions that update VMCS, EPT, Control Registers directly in the kernel. For the former ROP attack model, we do assume that the attacker can invoke these functions with ROP payload, we do not limit that how the attacker invokes these functions. For the latter ROP attack model, we emphasize that HyperPS would verify values updated to VMCS, EPT and Control Registers. We do not impose a lot restrictions on how the attacker tampers these data structures, we do care about the values updated to these data stucutures. Actually, we present a powerful adversary. In details, as depicted in Figure 3 in our paper, the EPT Management Component and the HyperPS Space Management Component would verify if the values that come from outside can be accepted or not. In other words, even if the attacker has tampered with the values of VMCS, EPT and CR3 though the ROP attack in the HostOS/Hypervisor Space, we can also guarantee the integrity of HyperPS Space and security-related data structures.

We do believe that we have not made the above clearly in submitted version, we have modified the Section 3.3 (threat model) and Section 4.1 in the revised version.

Q2: Please clarify if HyperPS uses binary inspection technology.
A2: Because of the different threat model, in this paper, we do not use binary inspection or VMI-like technologies. Binary inspection, such as VMI, is usually used for protection of virtual machine. In the VMI protection scenario, the VMI locates in the HostOS, which means the HostOS is completely trusted. Meanwhile, the adversary, in the VMI protection scenario, attempts to subvert the virtual machine he accesses to, but not other virtual machines locate at the same physical machine.

However, HyperPS assumes that the HostOS/Hypervisor is not trusted any more. HyperPS attempts to deprivilege some privileges that belong to HostOS to protect the virtual machines against the compromised HostOS. In our assumption, the attacker can deploy some virtual machines for future attacks. The attacker attempts to subvert other virtual machines locates in the same physical machine as the virtual machines the attacker deploys.

We believe that we not explain this point clearly, we do some modification in the Section 'Introduction' and Section 'Motivation' to make this point much more clearly. In the Section 'Introduction', we add a brief summary to the point for this work. We detail motivation for our work in the Section 'Motivation'.

Q3: The introduction section doesn't provide any hint as to how the mechanism of HyperPS actually works.
A3: In the introduction section of this revised manuscript, we have added some details on the work mechanism of HyperPS.

Q4: please detail the third and fourth attacks in the table in the evaluation section.
A4: In this paper, the third and the fourth attack are used to illustrate if HyperPS is immutable to HyperPS crash or HyperPS bypass attacks. Details have been listed in Section 6.1.2 'HyperPS Space Violation'.

Q5: about the SPEC CPU2006 workloads
A5: We indeed made writing mistakes in this section. Some items in SPEC CPU are going to test how the tested program performs in large and fluctuating memory footprint situations. We have corrected these mistakes.

Q6: about the performance cost of HyperBench
A6: Since we modified the HostOS, especially the functions about the most frequency-accessed data structures: VMCS and EPT, HyperPS would introduce performance loss with no doubt. The HyperBench is the performance evolution tool directly evaluating the virtualization capabilities about software and hardware platform without needing the workloads with varying number of vCPUs and memory size. More details about HyperBench can be found in the cited paper.

Q7: about the "sharedID" field and shared buffer.
A7: SharedID is used to tackle KSM, this is a technical problem, we explained how HyperPS uses SharedID to tackle KSM in Section 5.4 'KSM Handle'. However, we do believe that we should give more details in the section 'Design' to make this more clearly. We have modified the Section 4.3.2 'EPT Paging-structure Protection' to detail the use of SharedID. SharedBuffer mainly contains switch gate code, the entrance and exit address in and out to HyperPS Space. Details can be found in Section 5.2 Switch Gate.

Q8: the comparison about NOVA is not clear.

A8: We have rewritten the related content with Nova, and made it more clear about the difference between Nova and our HyperPS.

Q9: about some language/presentation mistakes.
A9: We have made many works to revise language mistakes and improve the presentation in the manuscript. Thank the reviewers for your careful reviews.