(1) viruses that have the ability to gain access to kernel privileges and perform operations at the root administrator level and (2) how the buffer overflow attack exploits the process stack. Additional requirements follow.  
**Topic 1:** introduce the topic and describe methods for gaining the kernel / root access. What are some approaches to defeat such attacks? List some well known viruses of this type.   
**Topic 2:** explain why a process needs to maintain a stack and how the buffer overflow attacks work to subvert the was the stack is used by the process. What are some approaches to defeat such attacks? List some well known viruses of this type.

Operating systems for mobile devices, desktop computers, and servers are all designed with the notion of multiple privilege levels for processes. A basic user account is given less access and rights to system files and programs. An administrator account is used for this higher level of access. Bugs in software can render a computer system vulnerable to being compromised, but also often exploited are conditions arising from mis-configured software and incorrect system settings. These vulnerabilities are what computer viruses can exploit after arriving on your system, commonly in the form of a malicious email attachment, an unvetted software download or infected physical media introduced to the system.

A worm is a virus that propagates on its own over the network. In 2001

On UNIX systems, ‘SUID (Set User ID) files execute with the privileges of the file owner, rather than the user, and a writable SUID file would essentially allow the user to execute any commands the owner of the file could execute simply by modifying the file.’ (Teodorczyk)

In 2016 a Linux vulnerability called ‘Dirty Cow’ was discovered, named such due to its exploiting a race condition in the ‘copy on write’ kernel functionality. This exploit affected Android mobile devices as well. The result of this programming oversight was that an unprivileged user could gain write access to read-only memory mappings and thus leverage access to protected system actions. The term ‘dirty bit’ refers to a bit that indicates if a particular block of memory has been modified. The fix in this case was to properly check the ‘dirty bit’ and introduce a new flag to mark the copy on write as complete.

In 2011 an exploit affecting Windows NT and derivatives was discovered, dubbed Dugu 2.0, the exploit employed a specially crafted TrueType font. Because ‘Windows executes TrueType font programs for rendering bitmaps in Ring 0’ (Wolf), the exploit allowed for full system access. Specifically, a bug in WIN32K.SYS results in not checking bounds when merging two bitmap font glyphs together. This allowed for hijacking an X86 OR instruction resulting in setting one-bits to arbitrary values.

Schauer, Renee C. “The Mechanisms and Effects of the Code Red Worm” SANS Institute InfoSec Reading Room, https://www.sans.org/reading-room/whitepapers/dlp/mechanisms-effects-code-red-worm-87

Teodorczyk, Marcin. “Understanding Privilege Escalation.” ADMIN Magazine, [www.admin-magazine.com/Articles/Understanding-Privilege-Escalation](http://www.admin-magazine.com/Articles/Understanding-Privilege-Escalation).

Rodrigo Freire & David Sirrine. “Understanding and mitigating the Dirty Cow Vulnerability” access.redhat.com/blogs/766093/posts/2757141

Wolf, Julia. “Windows Kernel TrueType Font Engine Vulnerability” https://cansecwest.com/slides/2013/Analysis%20of%20a%20Windows%20Kernel%20Vuln.pdf