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 a worm called ‘Code Red’ was discovered that remotely compromised Microsoft IIS web server software using a buffer overflow. The buffer overflow was in the form of repeated ‘N’ characters tacked on to the end of an HTTP GET request, followed by a payload of instructions that propagated the worm further. The exploit took advantage of improper bounds checking between the IIS server and Microsoft Indexing Service. The solution was to remove the indexing functionality if the server did not need it.

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) UNIX distributions take care in systems not being vulnerable to SUID attacks out of the box, but an oversight in setting SUID rights by a system administrator can allow for an attacker with local access to leverage root access.

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.

A process maintains a stack to compartmentalize data, providing a helpful programming interface to using functions. A stack allows for two operations, pushing and popping. Pushing puts a new value on top of the stack and popping removes the top value from the stack, presenting the next value in line. The stack data structure is inherently useful in the operations of functions returning and making calls to other functions. Whenever a function is called it is given a stack frame. The stack frame contains the address at which to resume execution after the function completes, the arguments given to the function, and space allocated for local variables.

A stack overflow occurs when reading data into a stack variable from standard input or the network and bounds checking not being used. When the capacity of the variable is exceeded memory is overwritten and can cause the program to crash. However, if this is done carefully, with some knowledge of the system being targeted, the return address value in the stack frame can be changed to the address that contains other overflow data that was sent by the attacker. This data can be in the form of executable code. This attack is possible because data on the stack is read from lower to higher addresses. Buffer overflows can be prevented by checking the amount of data to be stored in a variable before assignment. Modern CPUs and operating systems also allow the stack to be set as non-executable when not needed.

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

“How Bad Is Dirty COW?” The Linux Foundation, https://www.linuxfoundation.org/blog/2016/10/how-bad-is-dirty-cow/

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

“Avoiding Buffer Overruns” Windows Dev Center. https://docs.microsoft.com/en-us/windows/desktop/secbp/avoiding-buffer-overruns

“Elements of a process” https://www.bottomupcs.com/elements\_of\_a\_process.xhtml