CLAIMS

~~We claim:~~

1. A system comprising:

a processor;

a kernel-mode module which, when operated by the processor, performs:

creating a section object,

associating the section object with a dynamic-link library,

first providing the section object associated with the dynamic-link library to a hotpatch mechanism, the dynamic-link library including hotpatch data for a component loader associated with the hotpatch mechanism, the hotpatch data, when applied, modifying the component loader to cause the component loader to request execute rights for the section object,

setting an entrypoint of the dynamic-link library such that the dynamic-link library becomes executable, and

second providing the section object to the hotpatch mechanism to cause the component loader to execute the dynamic-link library; and

the dynamic-link library which, when executed, performs:

creating a user-mode process, thread, or held reference, and

after the creating, indicating to the component loader that the dynamic-link library does not include hotpatch data, causing the component loader to unload the dynamic-link library.

2. The system of claim 1, wherein, prior to the setting, the entrypoint of the dynamic-link library is set to zero.

3. The system of claim 1, wherein setting the entrypoint comprises setting the entrypoint in a program execution header of the dynamic-link library to a routine which, when encountered by the component loader, causes the component loader to execute the dynamic-link library.

4. A system comprising:

a processor;

a kernel-mode module which, when operated by the processor, provides an executable user-mode component to a hotpatch mechanism; and

the executable user-mode component including an entrypoint which, when the executable user-mode component is loaded by a component loader associated with the hotpatch mechanism, causes the executable user-mode component to be executed, the execution of the executable user-mode component including:

creating a user-mode process, thread, or held reference which is associated with the kernel-mode module, and

after the creating, indicating to the component loader that the executable user-mode component does not include hotpatch data, causing the component loader to unload the executable user-mode component.

5. The system of claim 4, wherein the executable user-mode component is a dynamic-link library.

6. The system of claim 4, wherein the executable user-mode component is provided to the hotpatch mechanism as a hotpatch for a running instance of a system module of a target process.

7. The system of claim 4, wherein the entrypoint is included in a program execution header of the executable user-mode component and is set to a routine which, when encountered by the component loader, causes the component loader to execute the executable user-mode component.

8. The system of claim 4, wherein the kernel-mode module is to create a image file-backed section object for the executable user-mode component and to provide the image file-backed section object to the hotpatch mechanism, the image file-backed section object being associated with execute rights.

9. The system of claim 4, wherein the kernel-mode module and executable user-mode component are associated with a security agent, and the user-mode process, thread, or held-reference provides user-mode capabilities to the security agent.

10. A computer-implemented method comprising:

first providing, by a kernel-mode module stored in memory of and executing on a computing device, a section object associated with a user-mode component to a hotpatch mechanism, the user-mode component including hotpatch data for a component loader associated with the hotpatch mechanism, the hotpatch data, when applied, modifying the component loader to cause the component loader to request execute rights for the section object;

setting, by the kernel-mode module executing on the computing device, an entrypoint of the user-mode component such that the user-mode component becomes an executable user-mode component; and

second providing, by the kernel-mode module executing on the computing device, the section object to the hotpatch mechanism to cause the component loader to execute the user-mode component, the executing the user-mode component including:

creating, by the user-mode component, a user-mode process, thread, or held reference, and

after the creating, indicating to the component loader, by the user-mode component, that the user-mode component does not include

hotpatch data, causing the component loader to unload the user-mode component.

11. The method of claim 10, wherein the user-mode component is a dynamic-link library.

12. The method of claim 10, further comprising creating the section object and associating the section object with the component.

13. The method of claim 12, wherein the section object is a page file-backed section object.

14. The method of claim 10, wherein the component loader requests the execute rights for the section object responsive to the second providing.

15. The method of claim 10, wherein, prior to the setting, the entrypoint of the component is set to zero.

16. The method of claim 10, wherein setting the entrypoint comprises setting the entrypoint in a program execution header of the user-mode component to a routine which, when encountered by the component loader, causes the component loader to execute the user-mode component.

17. The method of claim 10, wherein the hotpatch mechanism performs an atomic modification of at least a part of binary of the component loader with the hotpatch data.

18. The method of claim 10, wherein the indication that user-mode component does not include hotpatch data is provided regardless of whether the user-mode component includes hotpatch data.

19. The method of claim 10, wherein the user-mode component and the user-mode process, thread, or held reference are associated with a security exploit.

20. The method of claim 19, wherein a kernel-mode module of the security exploit performs the first providing, the setting, and the second providing.