Arbitrary Code Mitigations
and
Remote Procedure Code Reuse Attacks

Por Adrián Barreal

Programa STIC, Fundación Sadosky

Fundación SADOSKY

Contexto

- Foco en las mitigaciones que intentan prevenir la inyección y/o ejecución de código arbitrario.
- **Pregunta clave**: suponiendo que bloqueamos totalmente el código arbitrario, ¿qué impacto tiene esto en el costo de desarrollar exploits?

Motivación

- David Weston & Matt Miller. Presentation. Aug. 2016. "*Windows 10 Mitigation Improvements*."
- Matt Miller. Windows Blogs, Feb. 2017. "*Mitigating arbitrary native code execution in Microsoft Edge.*"
- David Weston & Matt Miller. Presentation. Mar. 2017. "*Microsoft's strategy and technology improvements towards mitigating arbitrary native code execution*."

Algunas nuevas mitigaciones contra el código arbitrario

- Arbitrary Code Guard (ACG).
- Code Integrity Guard (CIG).
- (No) Child Process Policy.

Distribución de costos

Mecanismo de explotación.	Específico y bajo nivel: mayor costo a largo plazo.
Payload inyectable, la "carga útil" del atacante.	Reutilizable y alto nivel: menor costo a largo plazo.

Algunas alternativas

Ivan Fratric. Bypassing Mitigations by Attacking JIT Server in Microsoft Edge. Google Project Zero, 2018.

- Corrupción de datos.
- Técnicas de reutilización de código.
- Usar al proceso vulnerable como agente interactivo desde el motor de scripting.

Algunas alternativas

Ivan Fratric. Bypassing Mitigations by Attacking JIT Server in Microsoft Edge. Google Project Zero, 2018.

- Corrupción de datos.
- Técnicas de reutilización de código.
- Usar al proceso vulnerable como agente interactivo desde el motor de scripting.

Algunas alternativas

Ivan Fratric. Bypassing Mitigations by Attacking JIT Server in Microsoft Edge. Google Project Zero, 2018.

- Corrupción de datos.
- Técnicas de reutilización de código.
- Usar al proceso vulnerable como agente interactivo desde el motor de scripting.

Payloads basados sólo en reutilización de código

- Tediosos de escribir.
- Payloads difíciles de mantener y de modificar.
- Payloads acoplados al software vulnerable.
- Mayor costo a largo plazo.

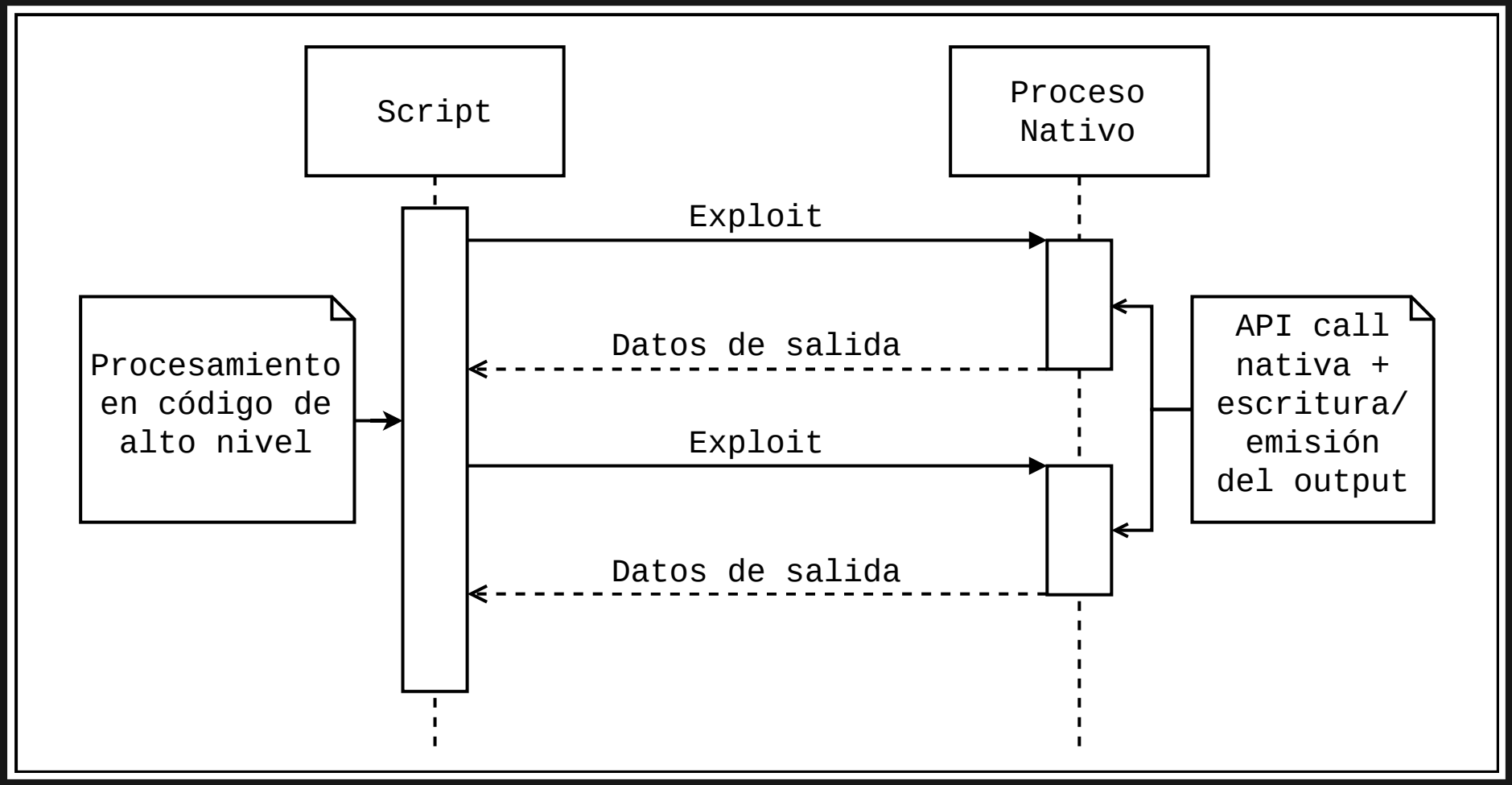
Idea sensata: forzar al atacante a depender solo de técnicas de reutilización de código.

Algunas alternativas

Ivan Fratric. Bypassing Mitigations by Attacking JIT Server in Microsoft Edge. Google Project Zero, 2018.

- Corrupción de datos.
- Técnicas de reutilización de código.
- Usar al proceso vulnerable como agente interactivo desde el motor de scripting.

Interacción con el motor de scripting

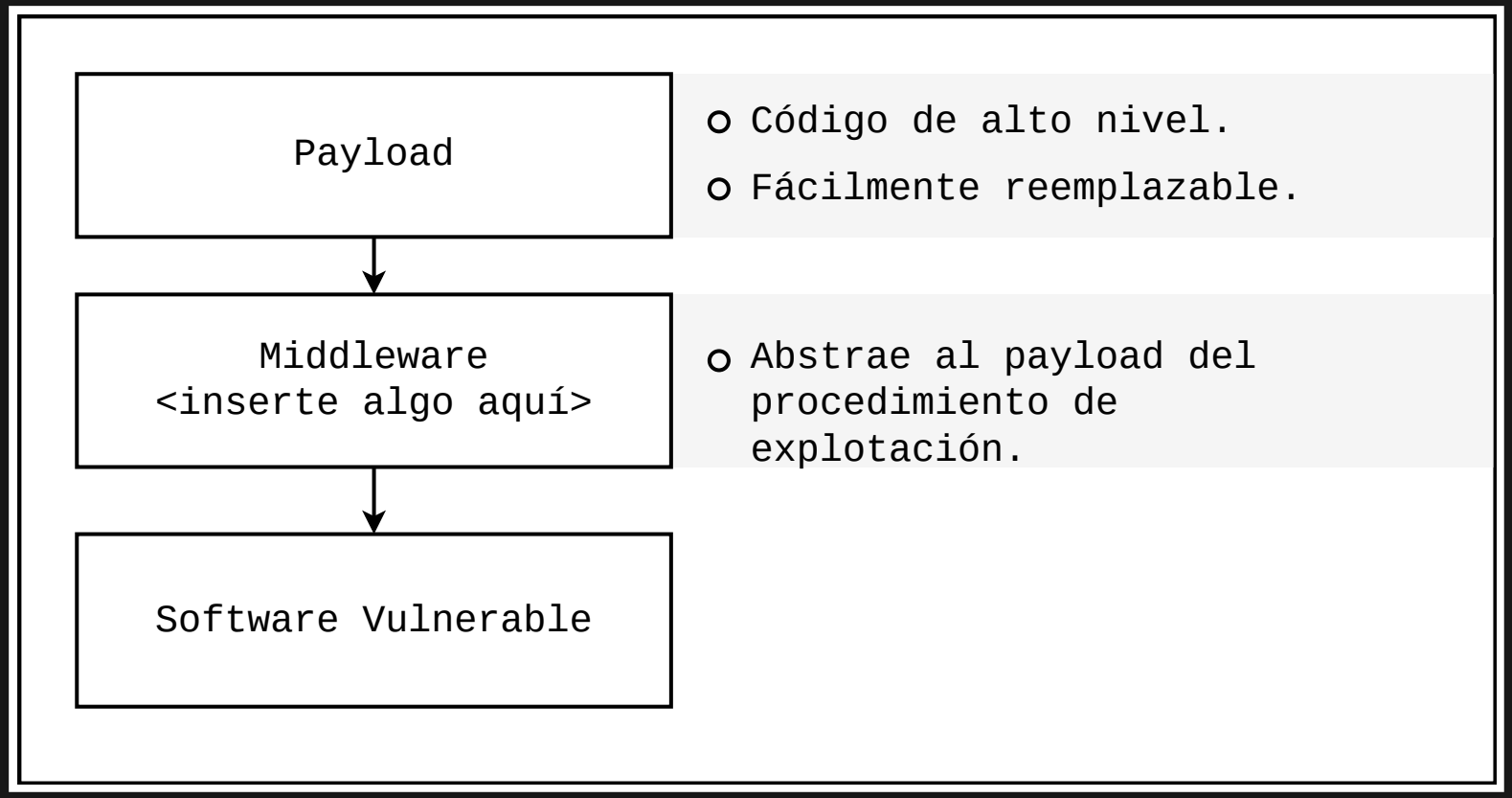


Interacción con el motor de scripting

- **Ventaja**: Payloads reutilizables, fáciles de mantener, fáciles de desarrollar.
  Bajo costo a largo plazo.
- **Desventaja**: ¿Requiere un motor de scripting?

Remote Procedure Code Reuse Attacks (RPCRAs)

Idea fundamental



Esencialmente...

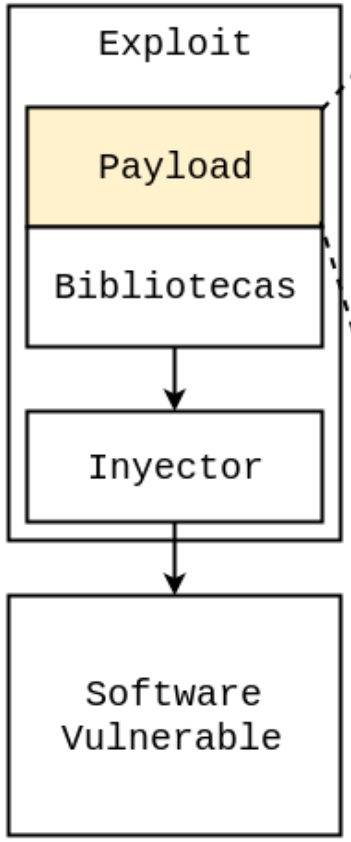
Maximiliano Caceres. Syscall Proxying - Simulating remote execution. Core Security. 2002.

... sin inyectar código arbitrario.

Demo

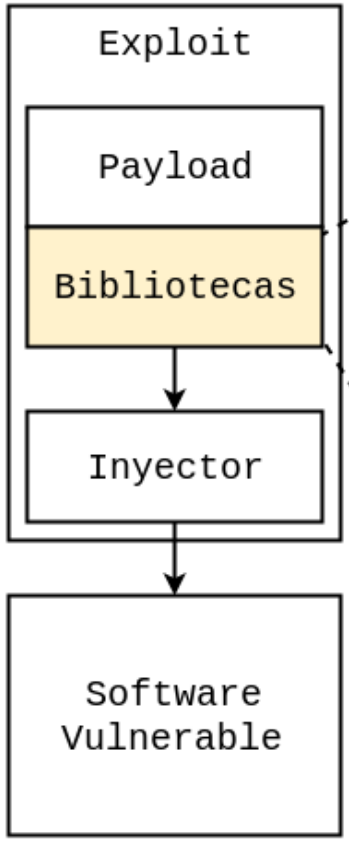
Task Scheduler remoto con ROP

Arquitectura



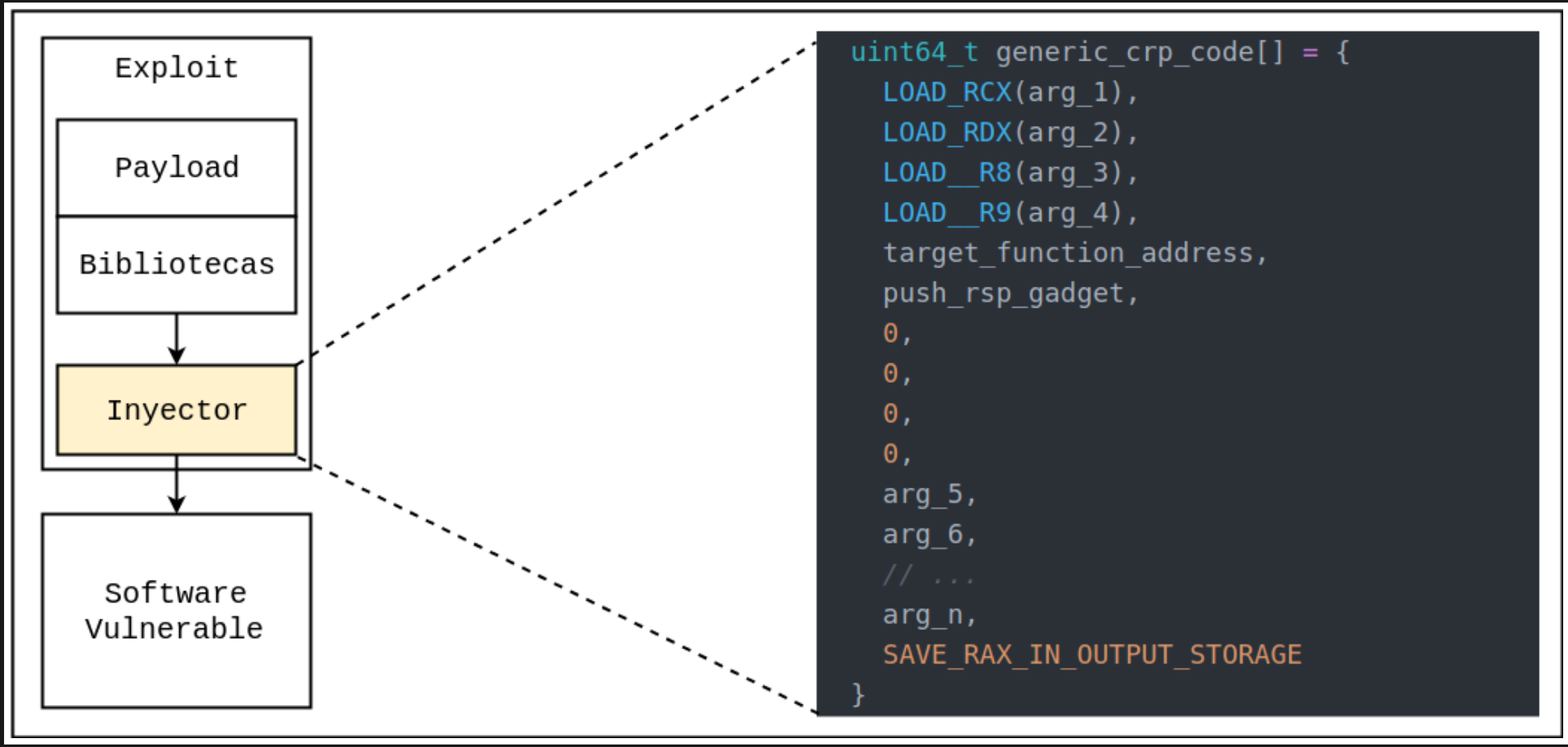

```
remote_GetUserNameW(&task_xml[XML_USERNAME_INDEX], &wlen);

task_xml[XML_USERNAME_INDEX + wlen - 1] = L' ';

remote_GetSchedBindingHandle(&r_binding_handle);
remote_RegisterTask(r_binding_handle, task_path, task_xml);
```

Arquitectura

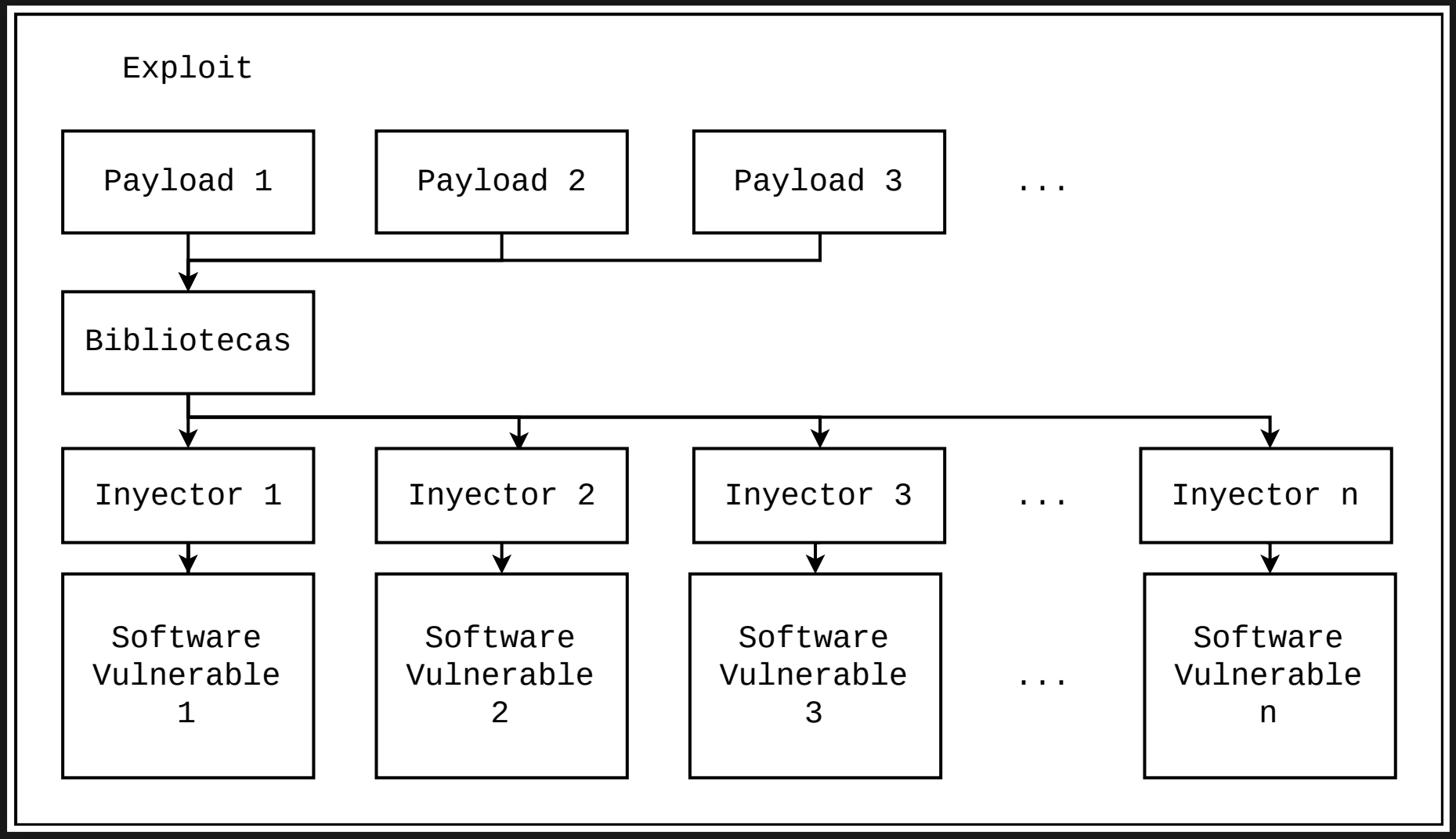



```
remote_find_and_call(
    "advapi32.dll",
    "GetUserNameW",
    0,                      // data
    0,                      // data_size
    &out,                   // local output buffer
    2,                      // argc
    r_user_name,            // arg1 (rcx)
    r_user_name_length); // arg2 (rdx)
```

Arquitectura

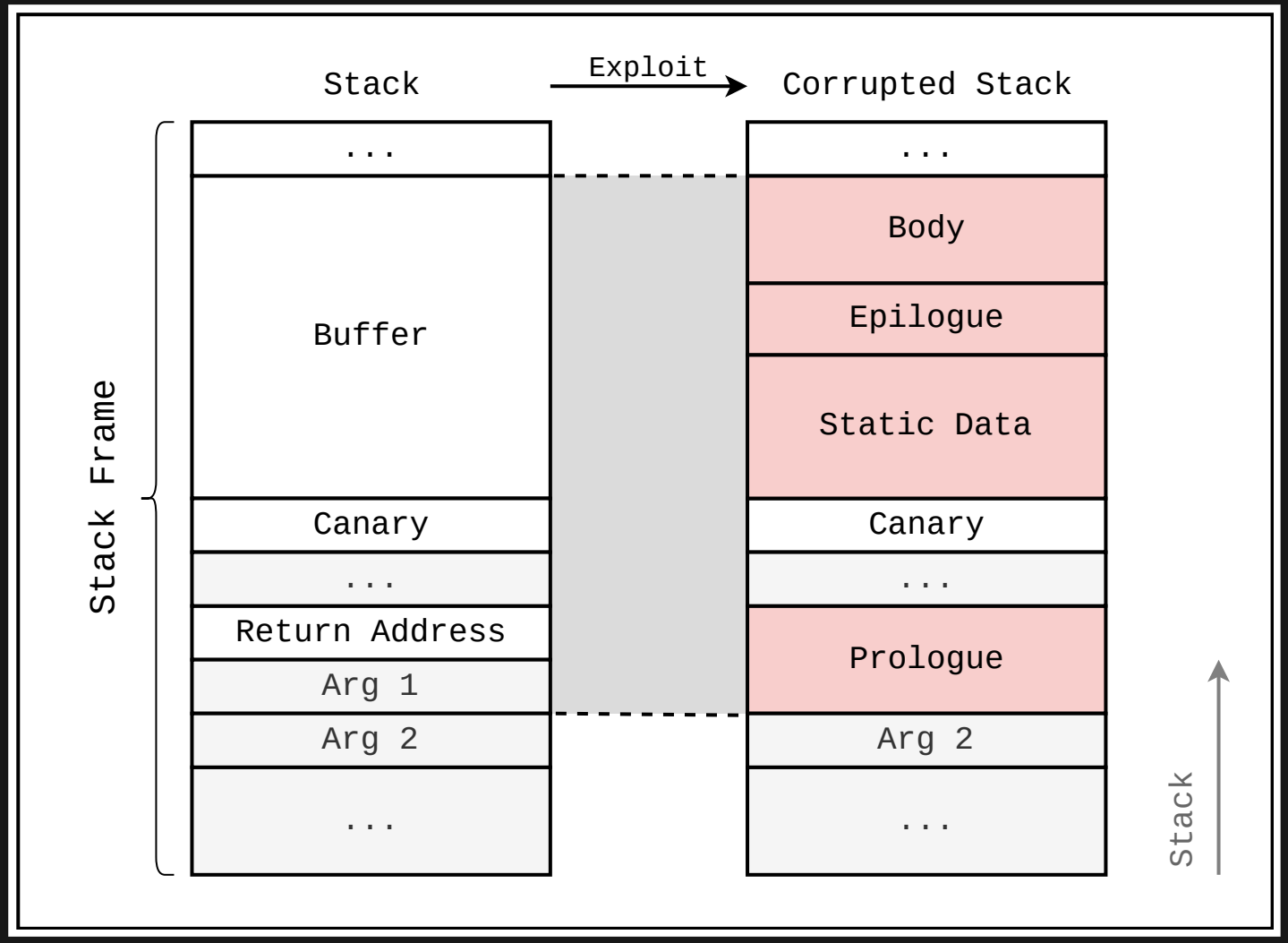


Interfaz común



¿Qué tan difícil es implementar el inyector?

Ejemplo básico



Ventajas de la técnica

- Payloads reutilizables.
- Código modular, extensible y de alto nivel.
- Cadenas de ROP cortas, simples y lineales.
- Menor costo a largo plazo.

Conclusiones y consideraciones adicionales

- Los payloads sofisticados todavía son prácticos.
- Escalando privilegios, los viejos payloads inyectables todavía sirven.
- Importante limitar privilegios y aislar procesos.
- Idealmente prevenir la corrupción de memoria o el control del flujo de ejecución en primer lugar.

Links

- Pruebas de concepto y documentación adicional: https://github.com/programa-stic/remote-procedure-code-reuse-attacks
- Sitio de STIC (Fundación Sadosky): http://www.fundacionsadosky.org.ar/programas/seguridad-en-tic/

Gracias

- Pruebas de concepto y documentación adicional: https://github.com/programa-stic/remote-procedure-code-reuse-attacks
- Sitio de STIC (Fundación Sadosky): http://www.fundacionsadosky.org.ar/programas/seguridad-en-tic/