ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

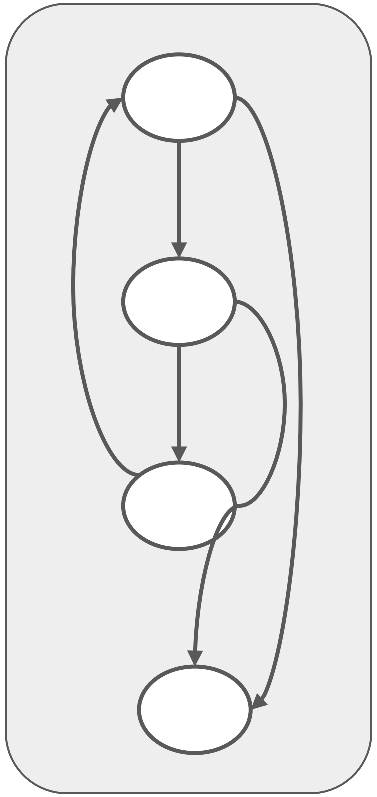
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

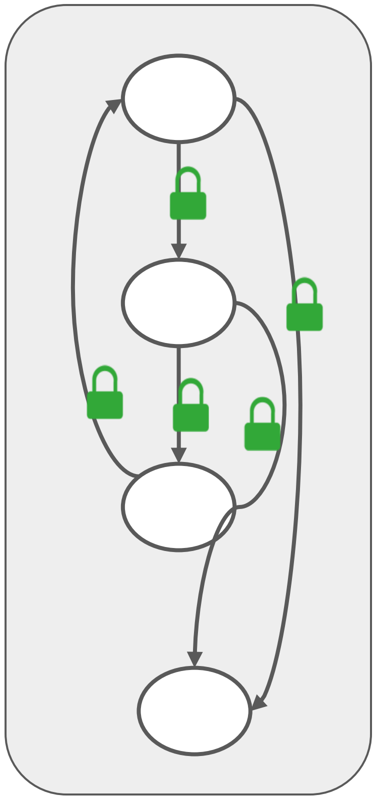
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

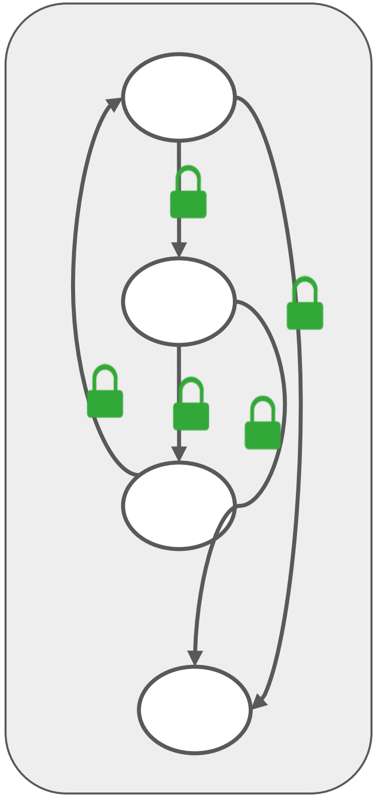
Practical Data-Only Attack Generation

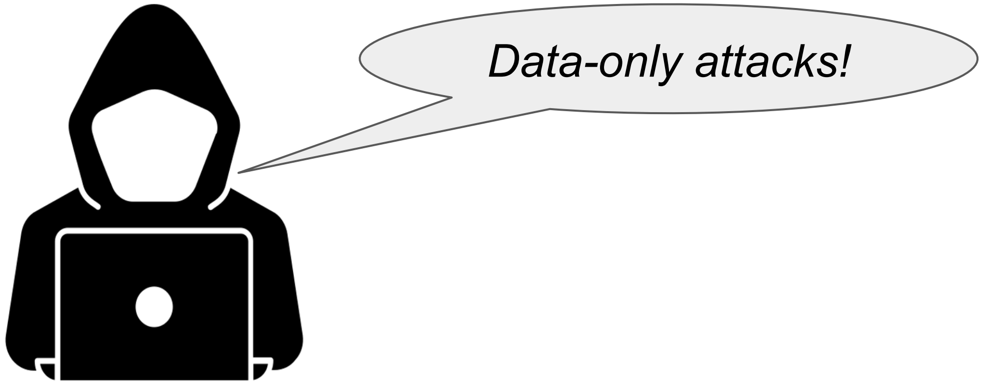
Brian Johannesmeyer, Asia Slowinska, Herbert Bos, Cristiano Giuffrida

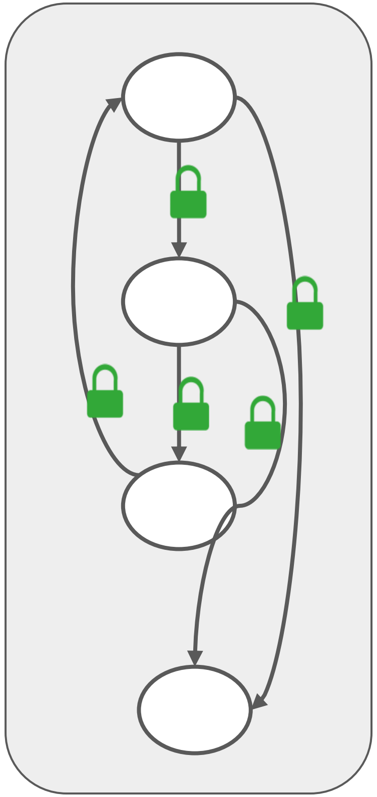
VU VRIJE UNIVERSITEIT AMSTERDAM



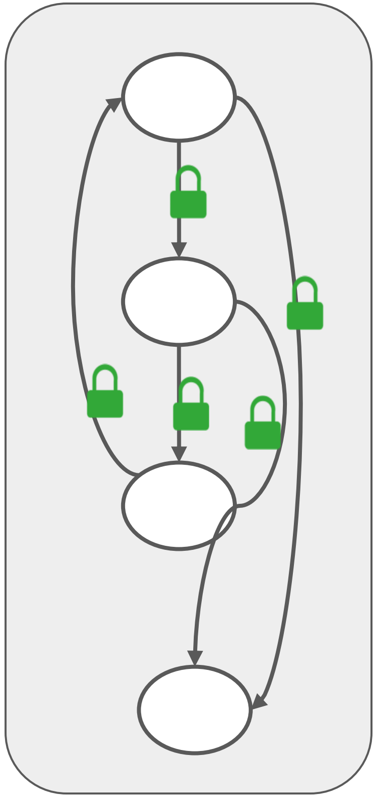




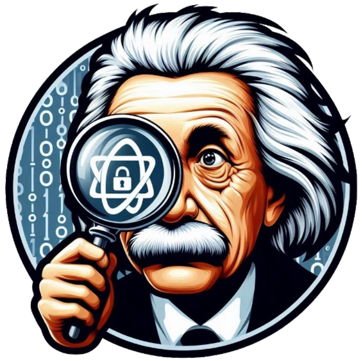

Data-only attacks!

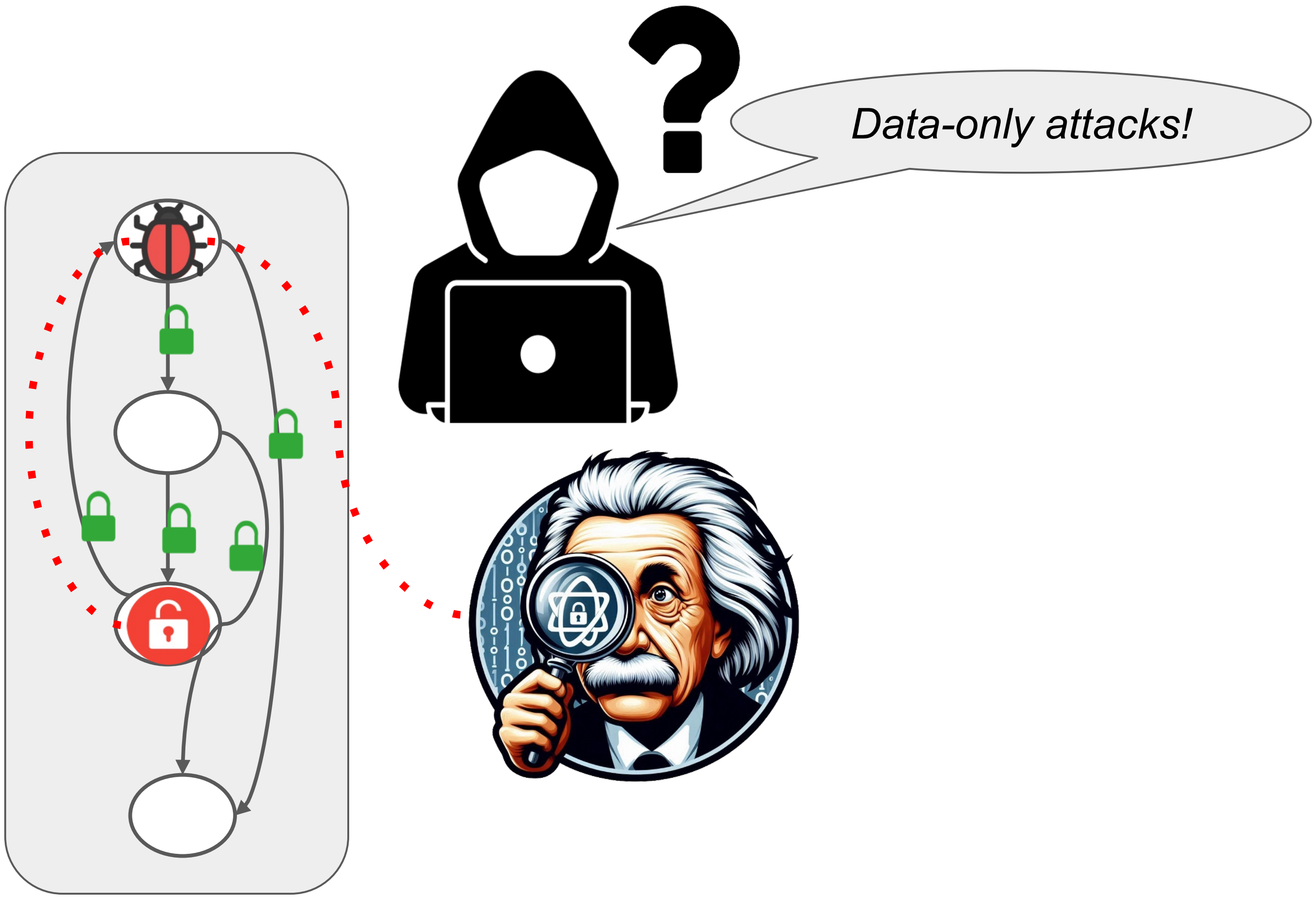


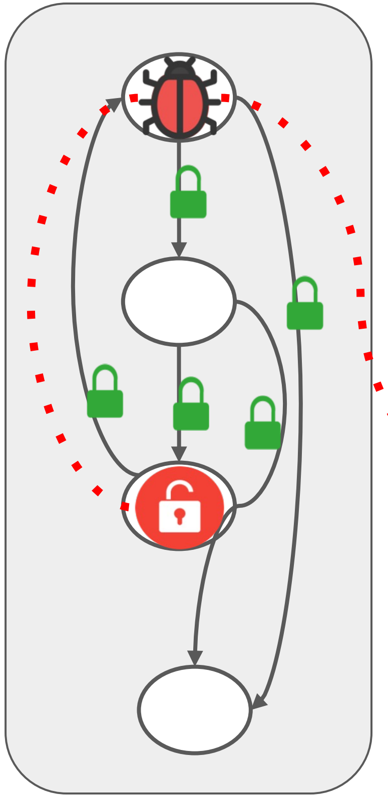
Data-only attacks!

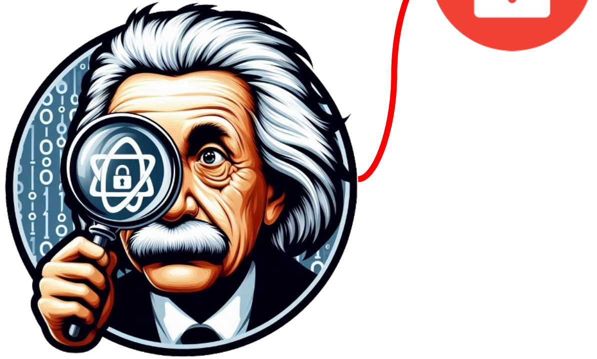


Data-only attacks!

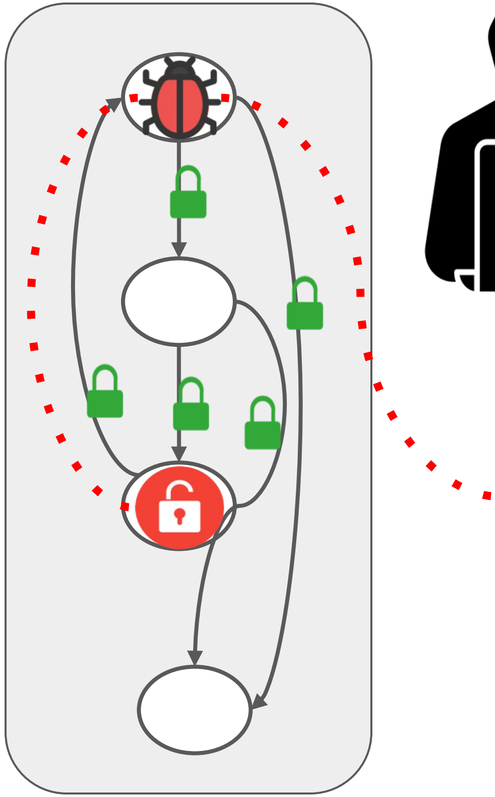


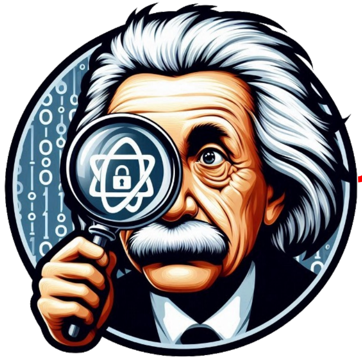

Data-only attacks!





*Builds **hundreds of exploits** against popular web servers*

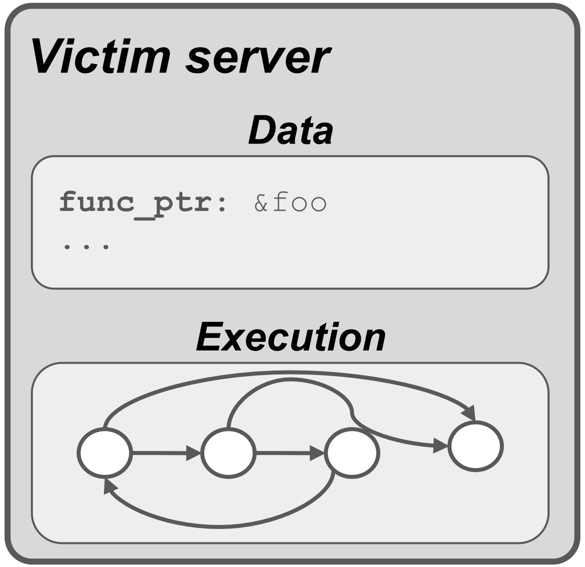




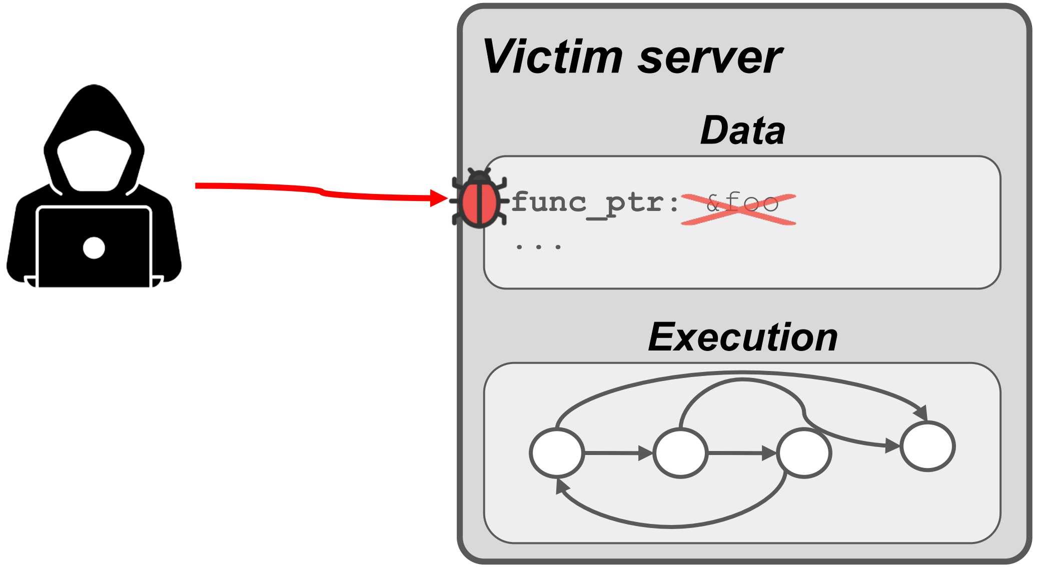
*Builds **hundreds of exploits** against popular web servers*

*Calls upon researchers and vendors to **rethink mitigation strategies***

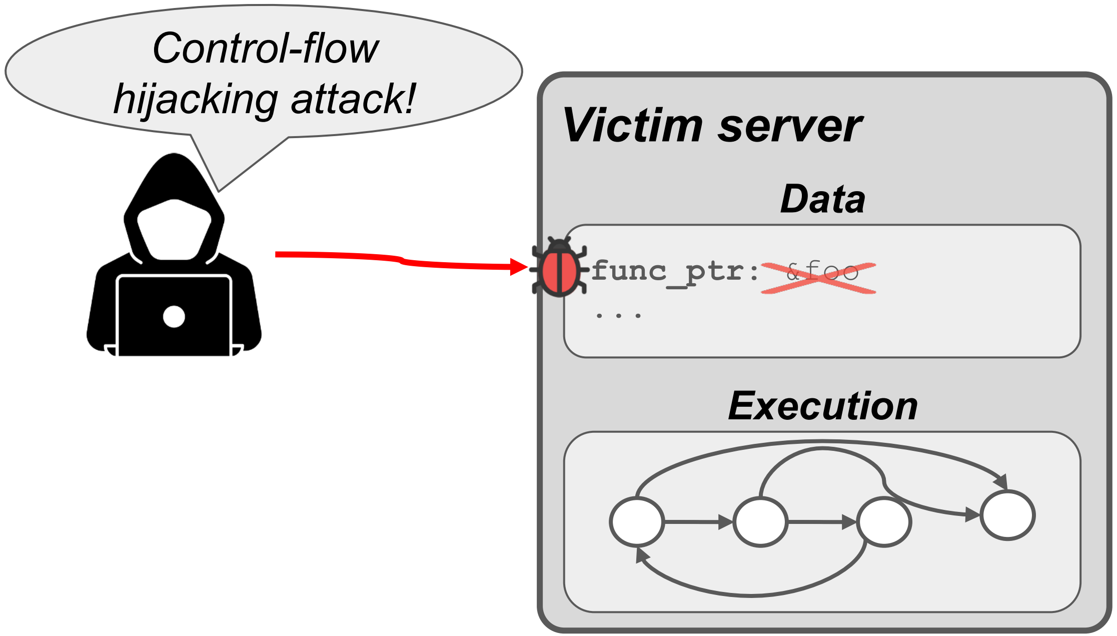
What is a data-only attack?



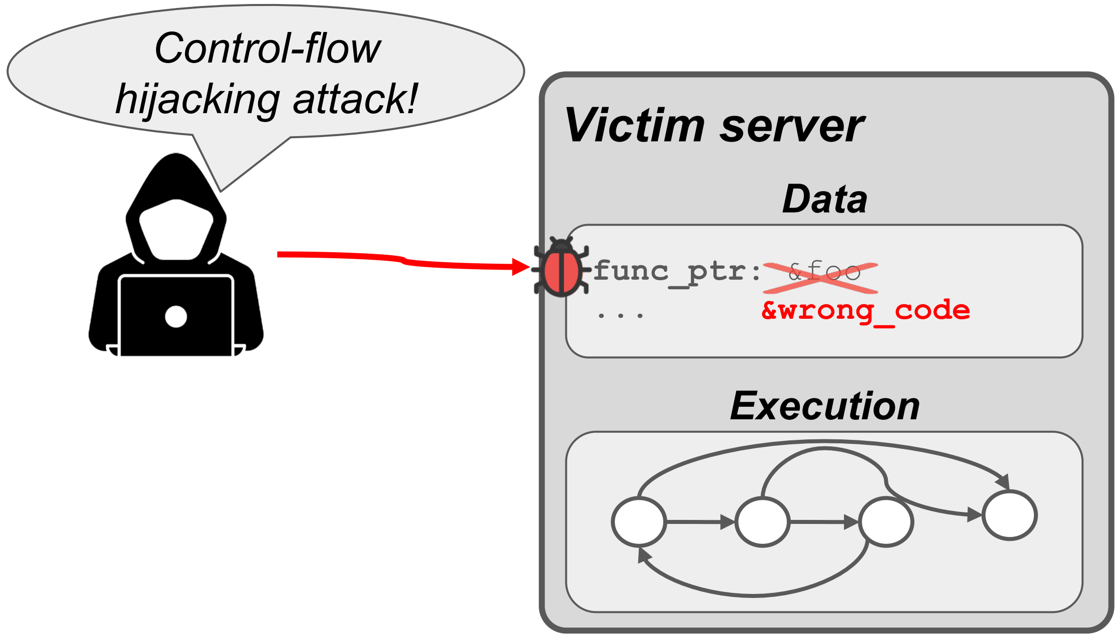
What is a data-only attack?



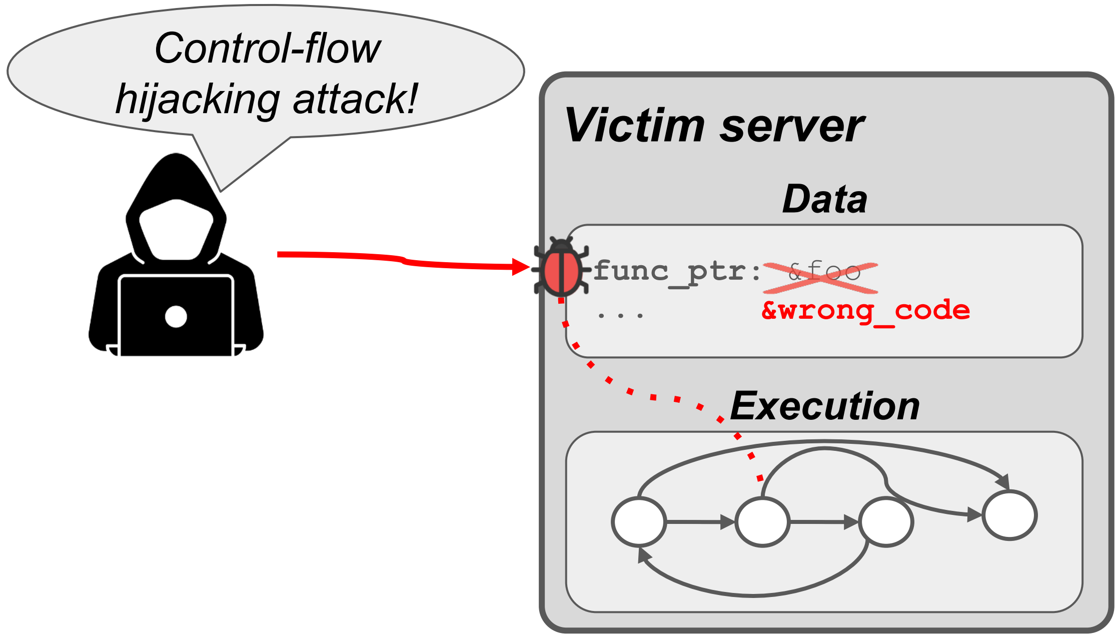
What is a data-only attack?



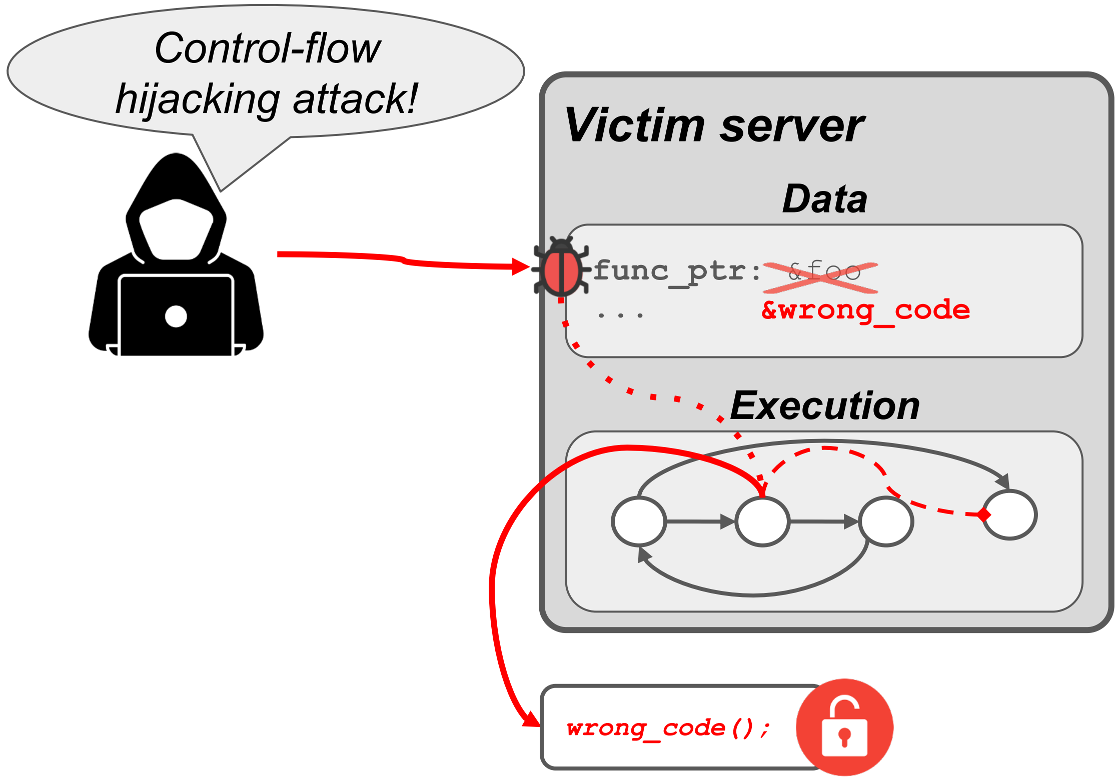
What is a data-only attack?



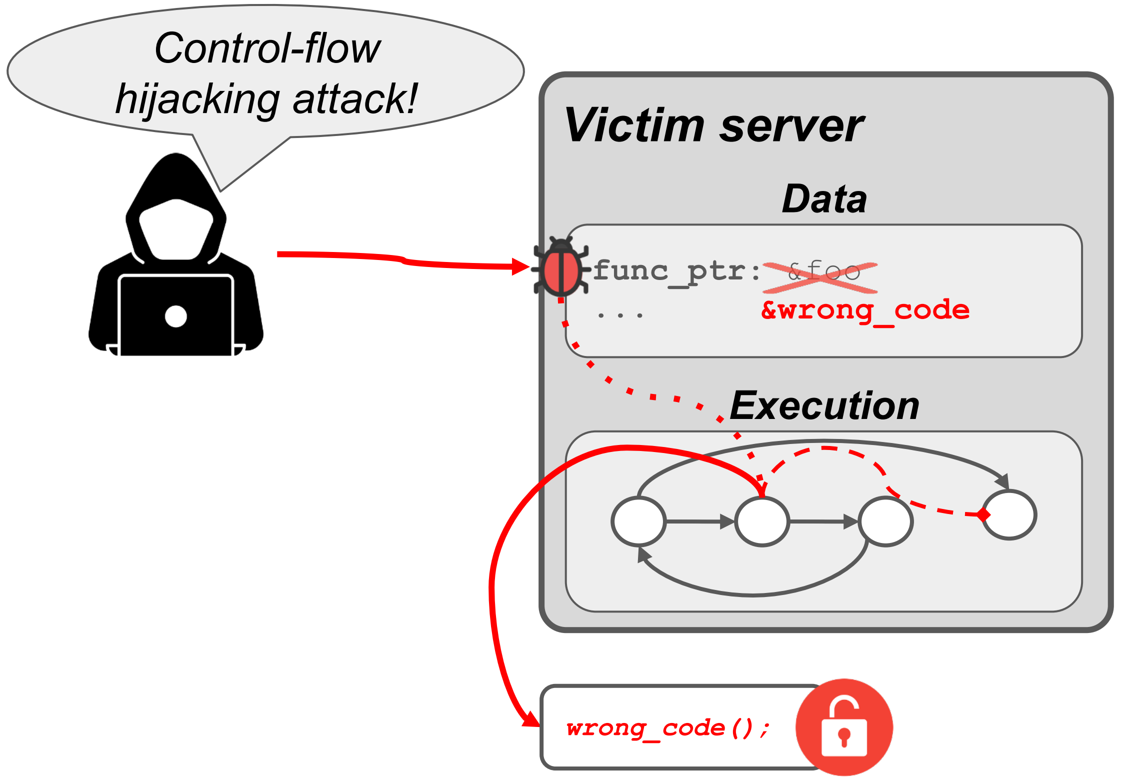
What is a data-only attack?

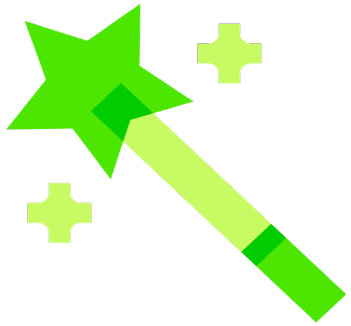


What is a data-only attack?

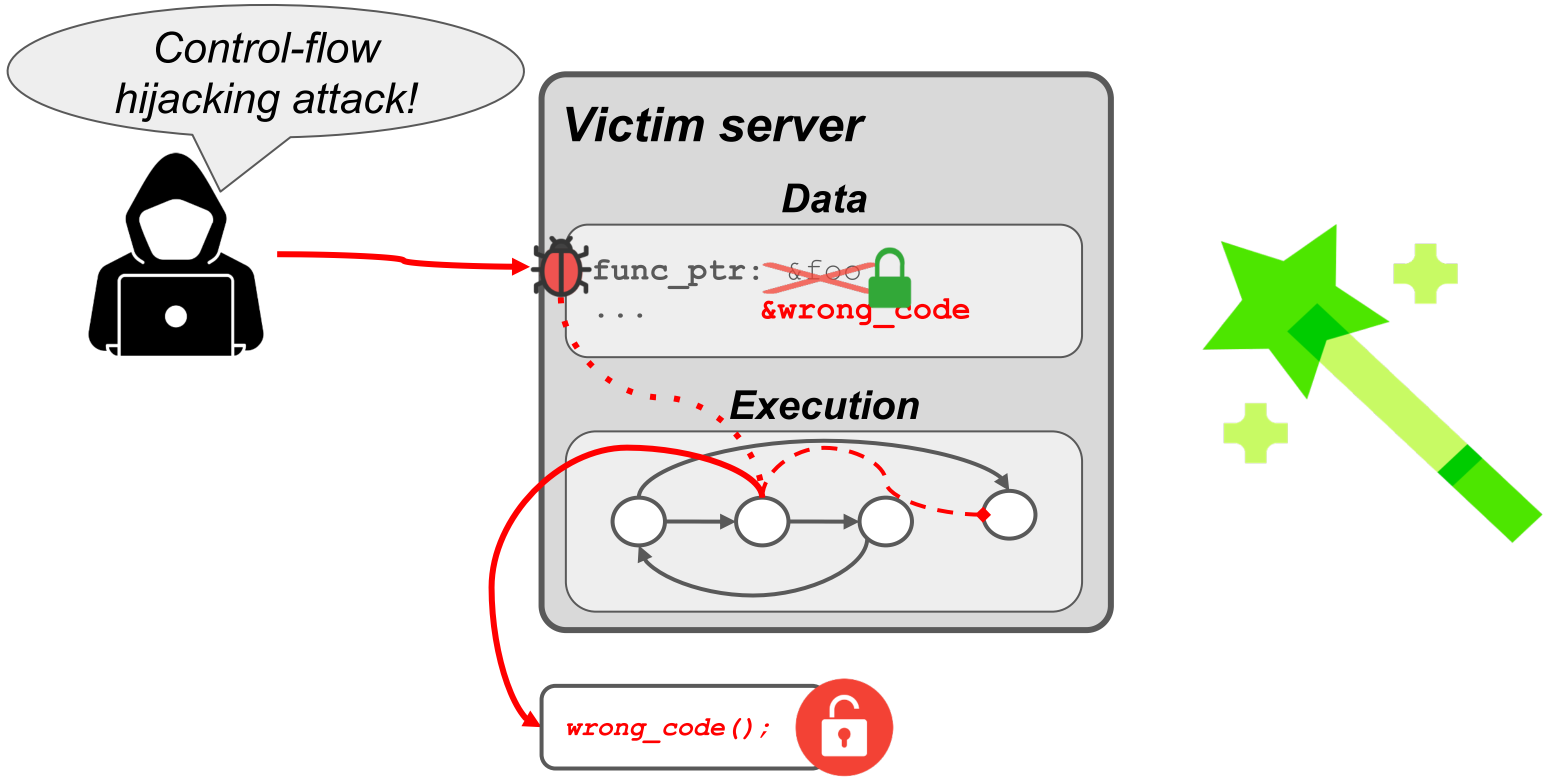


What is a data-only attack?

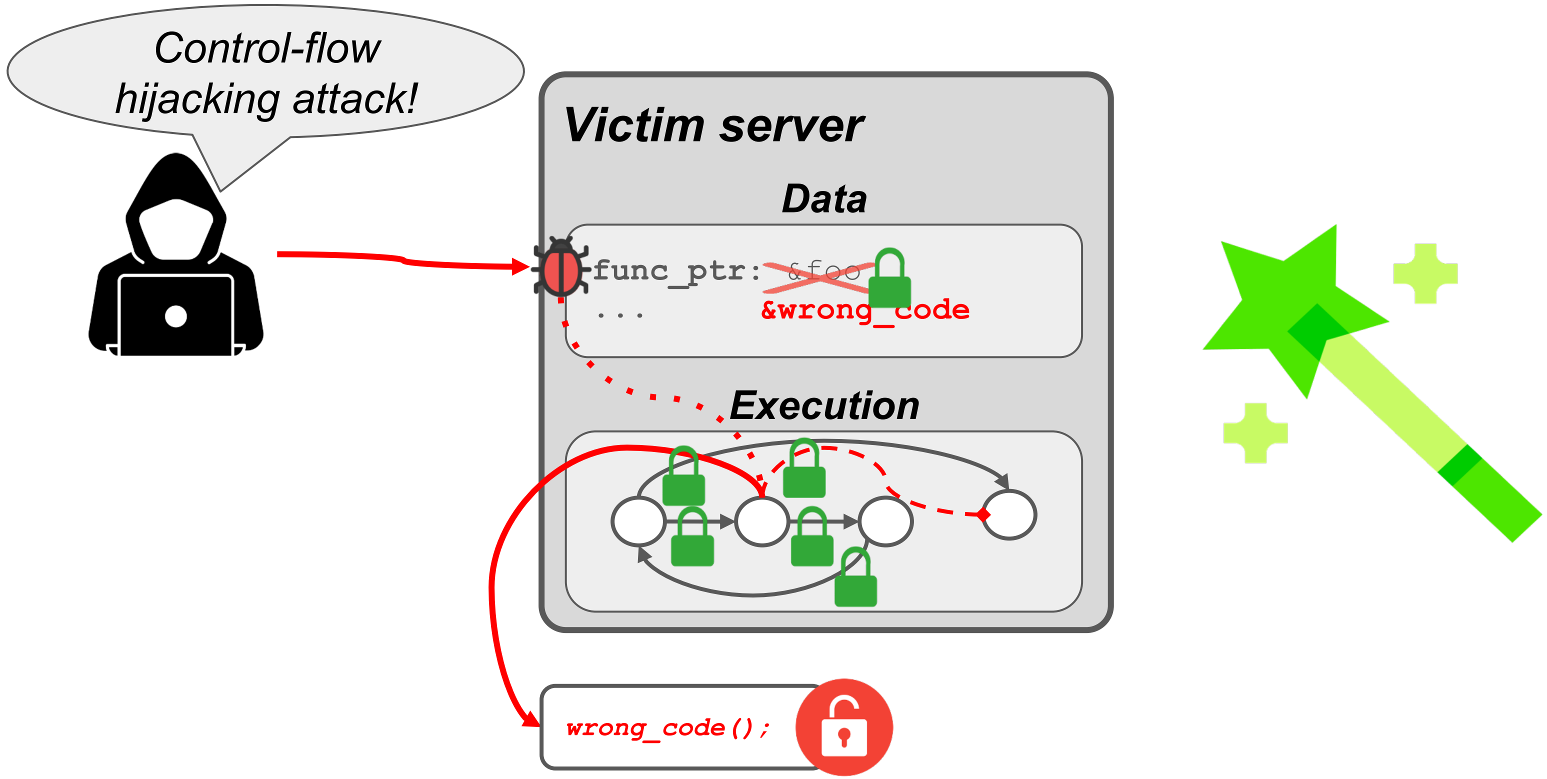




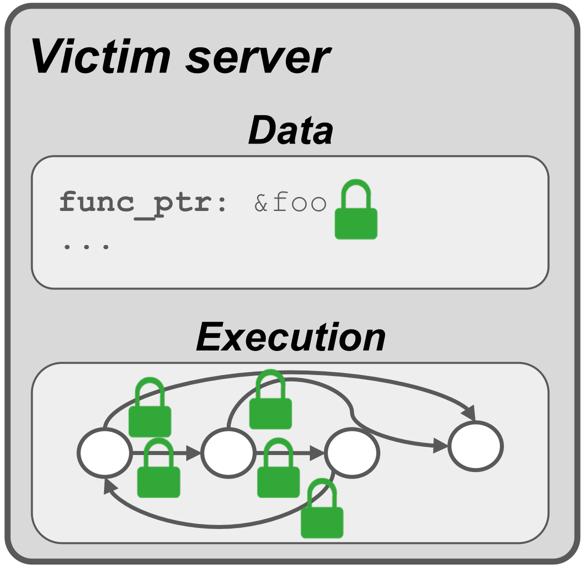
What is a data-only attack?



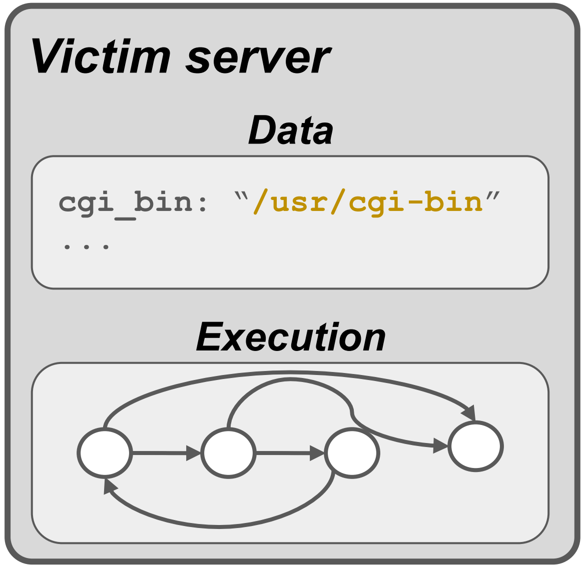
What is a data-only attack?



What is a data-only attack?

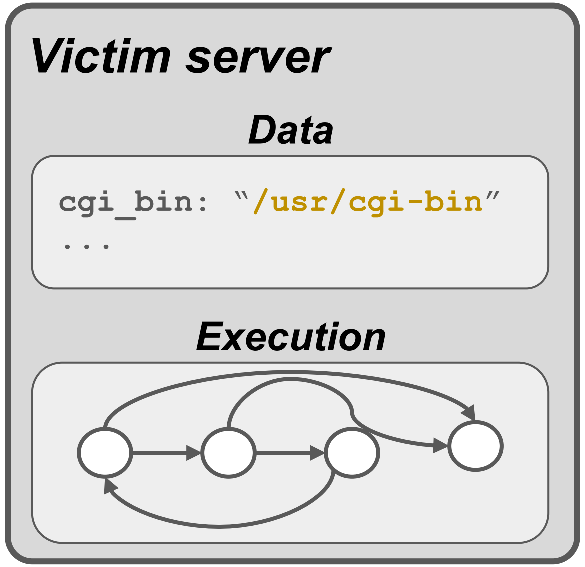


What is a data-only attack?



```
cgi_bin: "/usr/cgi-bin"
...
```

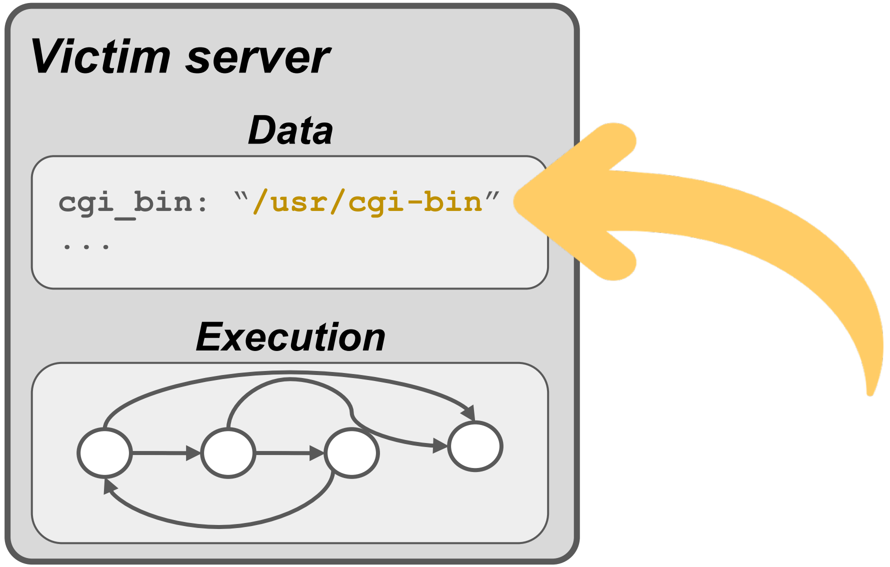
What is a data-only attack?



```
cgi_bin: "/usr/cgi-bin"
...
```

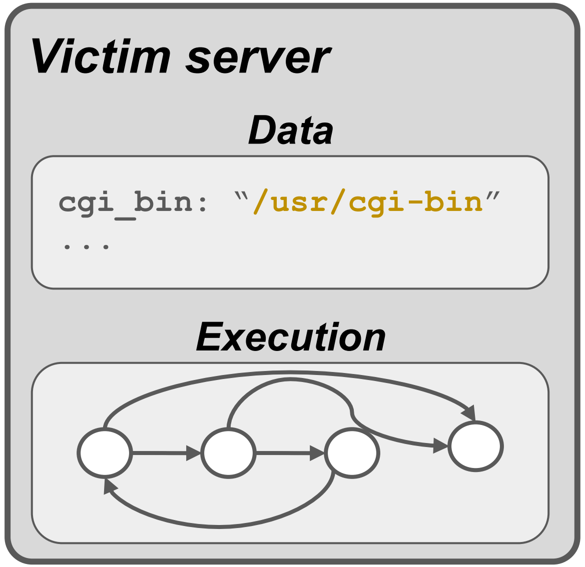
[S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer, "Non-Control-Data Attacks Are Realistic Threats," in USENIX Security, 2005.]

What is a data-only attack?



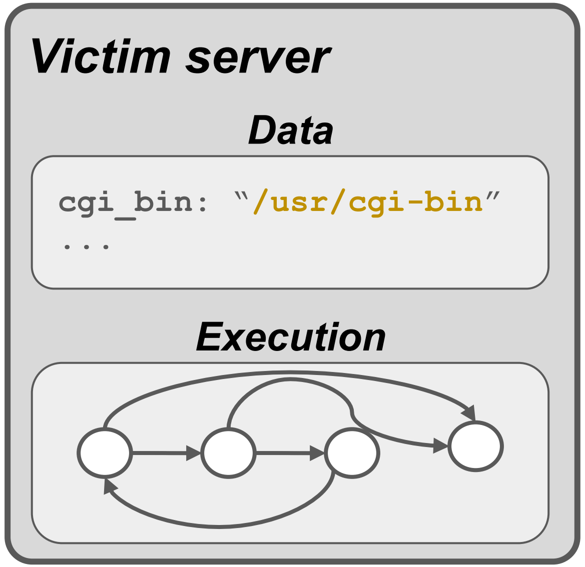
```
cgi_bin: "/usr/cgi-bin"
...
```

What is a data-only attack?



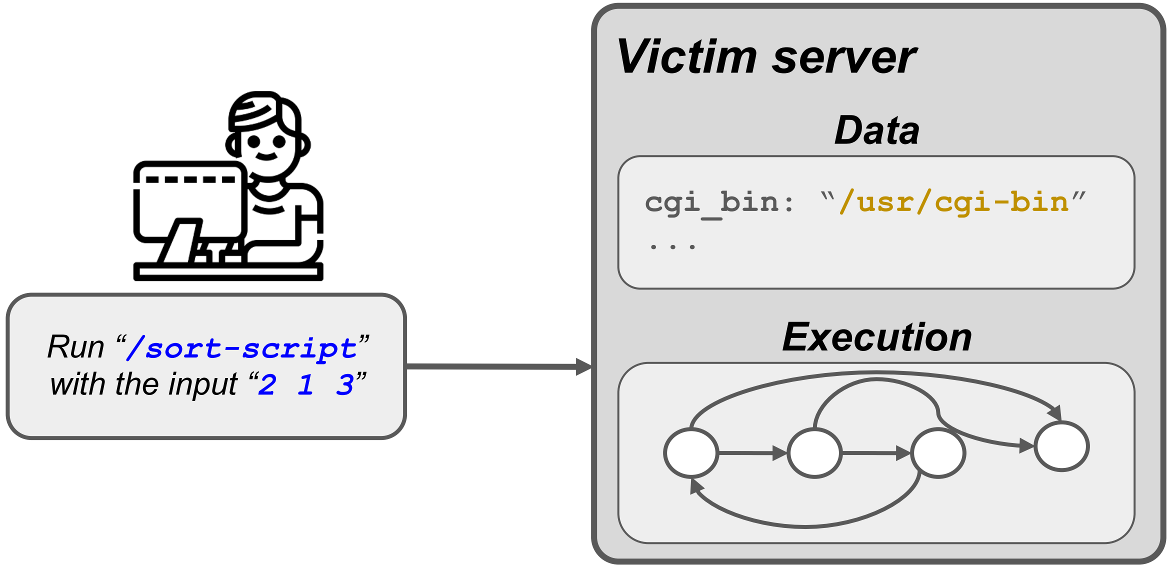
```
cgi_bin: "/usr/cgi-bin"
...
```

What is a data-only attack?



```
cgi_bin: "/usr/cgi-bin"
...
```

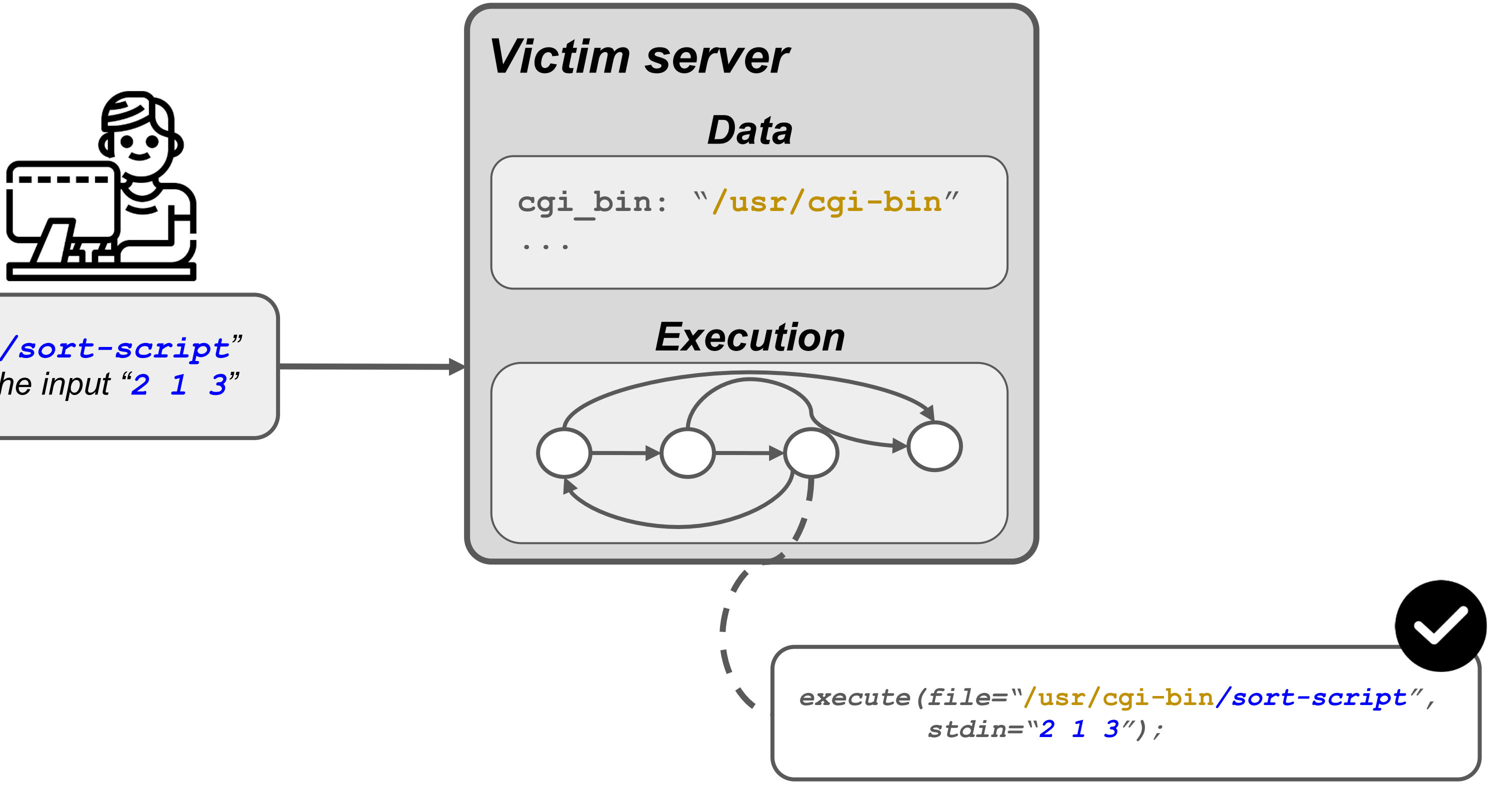
What is a data-only attack?



```
cgi_bin: "/usr/cgi-bin"
...
```

[S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer, "Non-Control-Data Attacks Are Realistic Threats," in USENIX Security, 2005.]

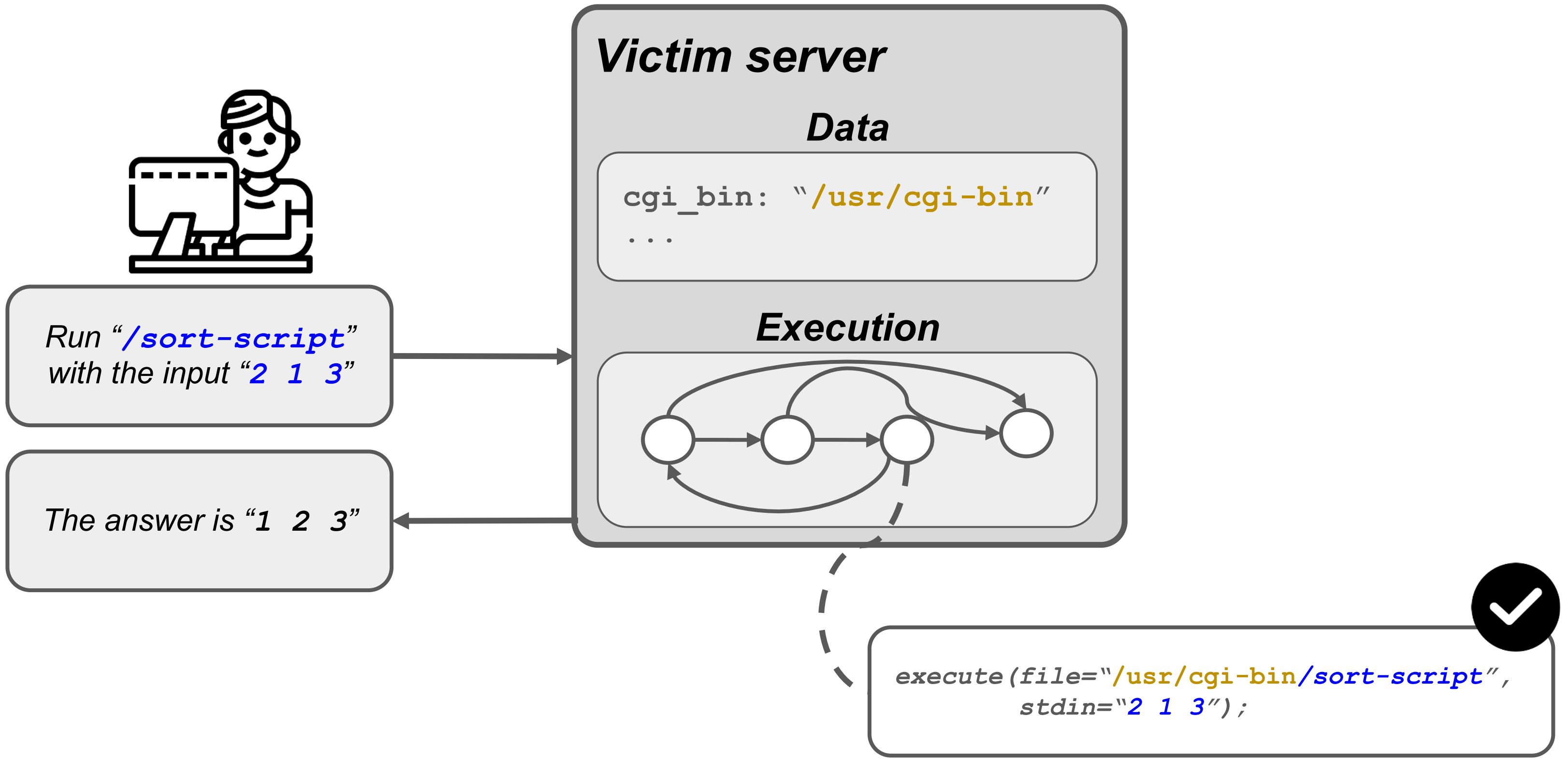
What is a data-only attack?



```
cgi_bin: "/usr/cgi-bin"
...
```

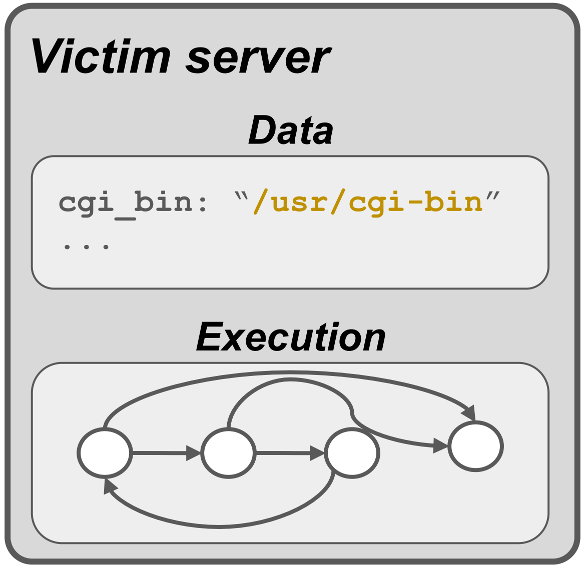
```
execute(file="/usr/cgi-bin/sort-script",
        stdin="2 1 3");
```

What is a data-only attack?



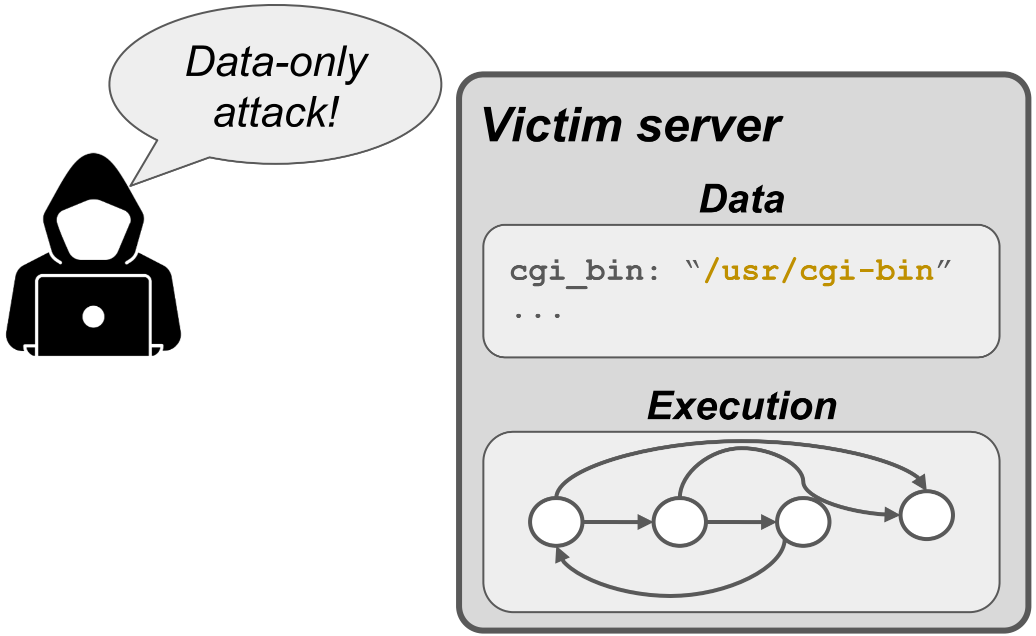
```
execute(file="/usr/cgi-bin/sort-script",
        stdin="2 1 3");
```

What is a data-only attack?

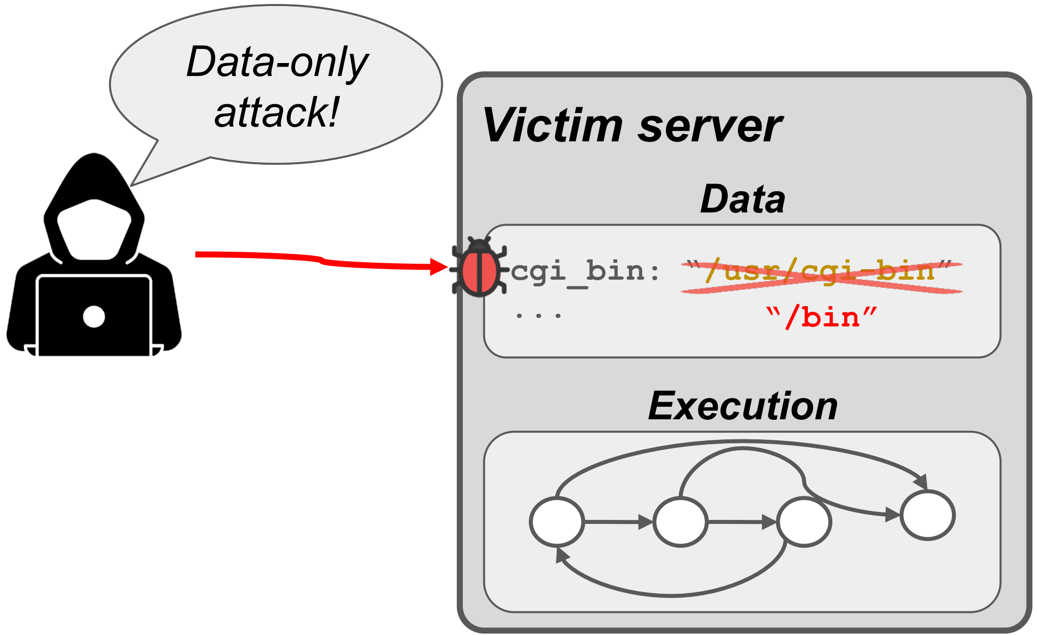


[S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer, "Non-Control-Data Attacks Are Realistic Threats," in USENIX Security, 2005.]

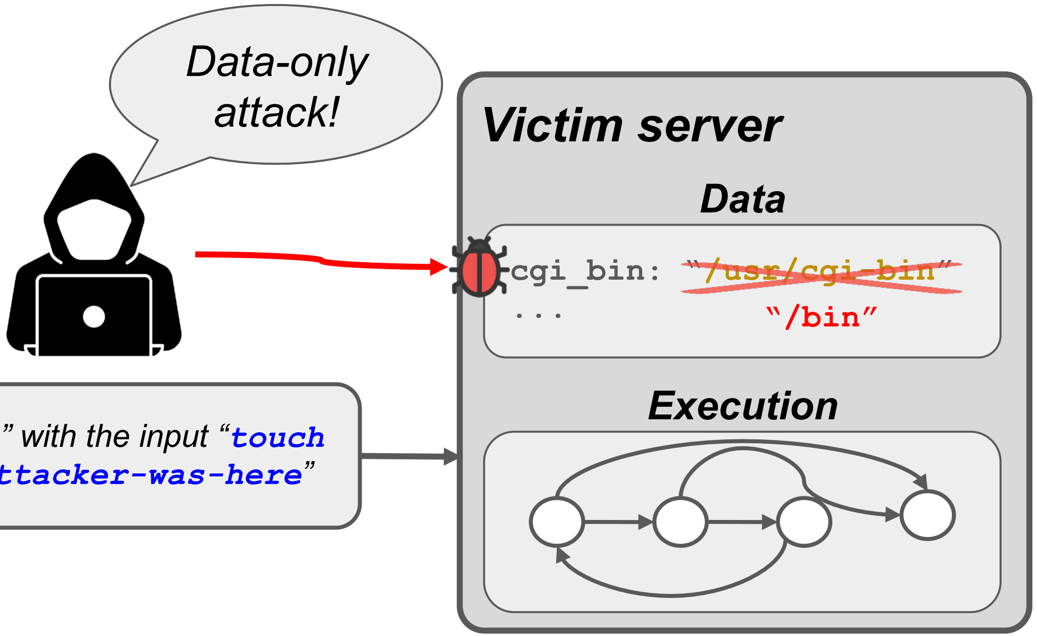
What is a data-only attack?



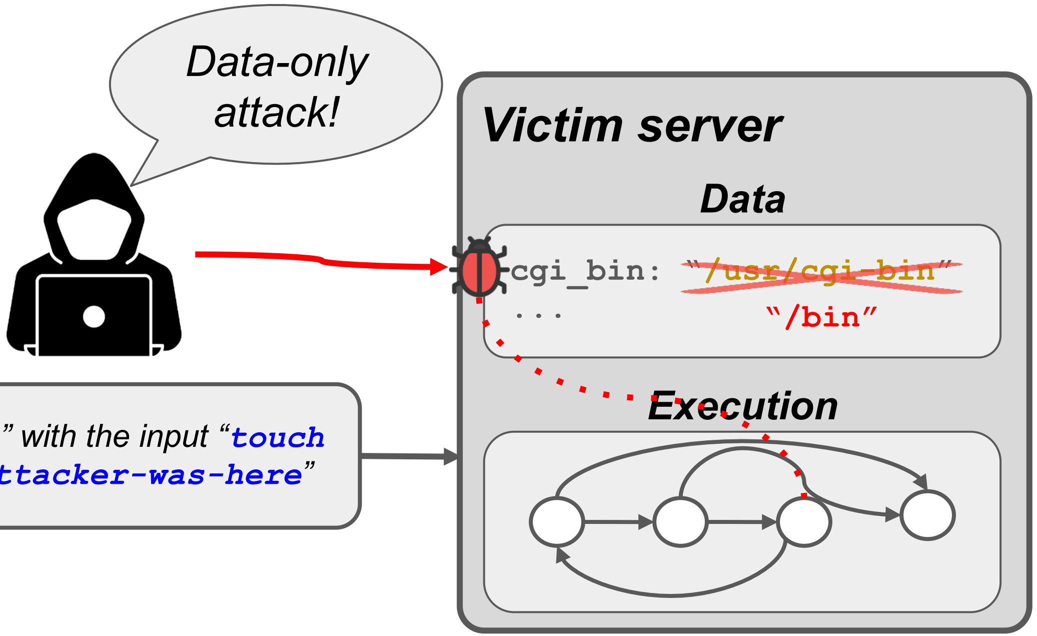
What is a data-only attack?



What is a data-only attack?

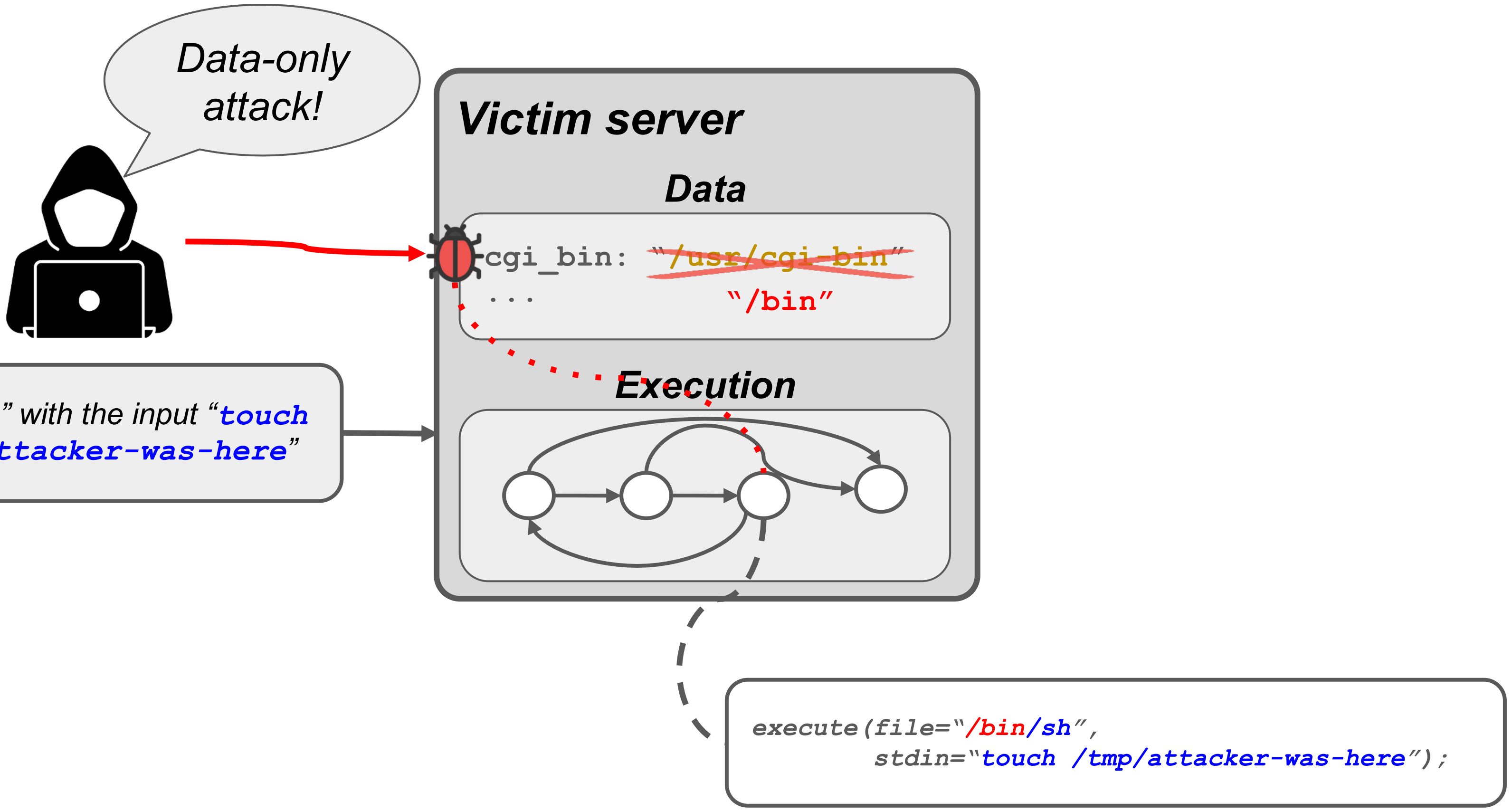

Data-only attack!
Victim server
Data
cgi_bin: ~~"/usr/cgi-bin"~~
"/bin"
...
Execution
Run "/sh" with the input "touch /tmp/attacker-was-here"

What is a data-only attack?



*Run "**/sh**" with the input "**touch /tmp/attacker-was-here**"*

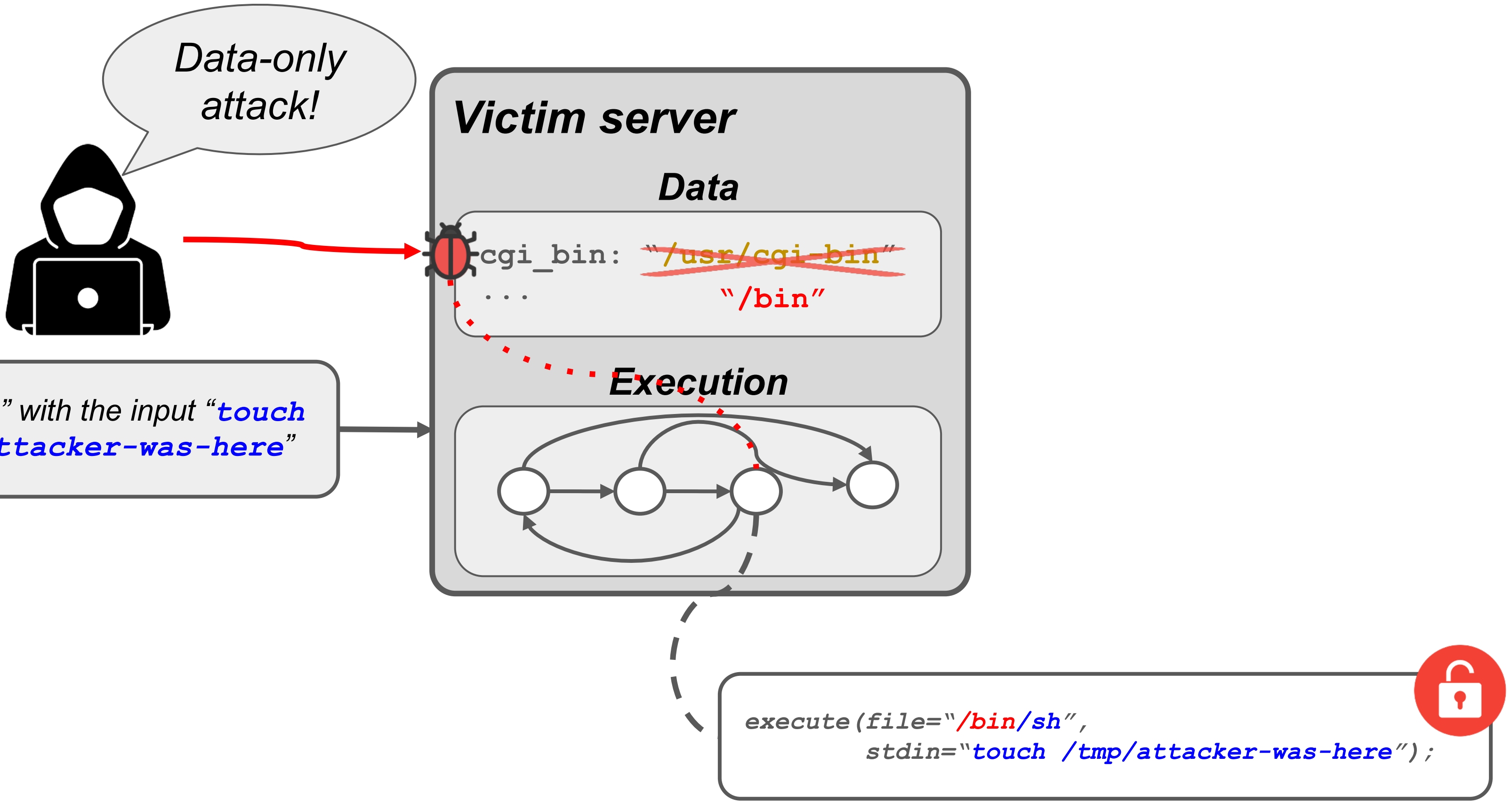
What is a data-only attack?



*Run "**/sh**" with the input "**touch /tmp/attacker-was-here**"*

```
execute(file="/bin/sh",
        stdin="touch /tmp/attacker-was-here");
```

What is a data-only attack?



Run "/sh" with the input "touch /tmp/attacker-was-here"

```
execute(file="/bin/sh",
        stdin="touch /tmp/attacker-was-here");
```

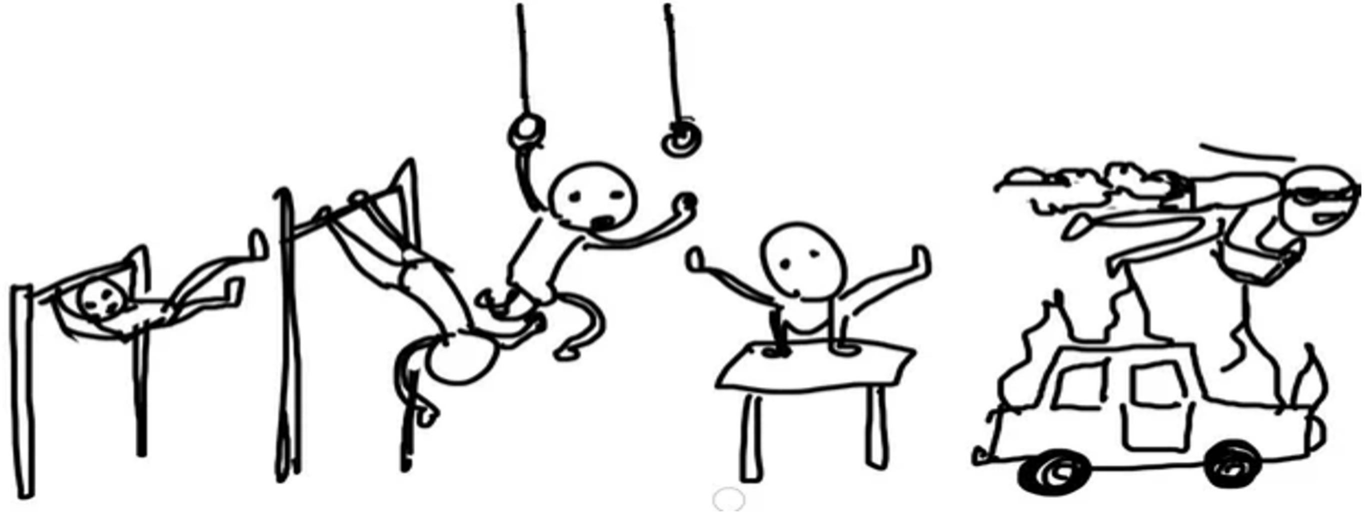
The attack does not corrupt the *victim's control flow*.

The attack does not corrupt the *victim's control flow*.

The attack does not corrupt the *victim’s control flow*.

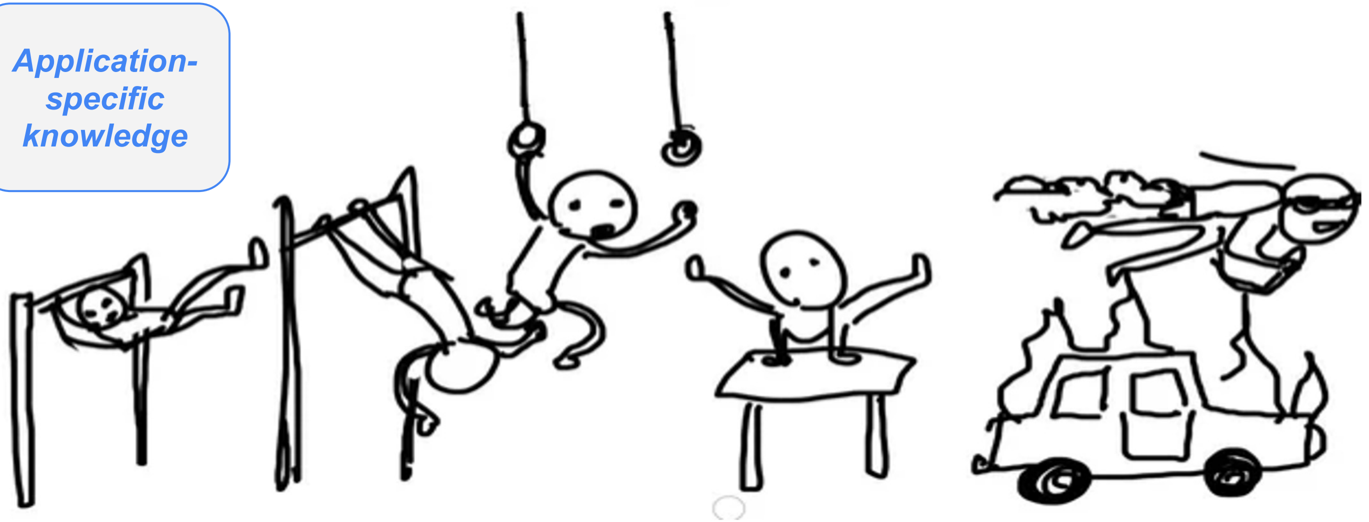
It only corrupts ***its function arguments***.

Why data-only attacks are **considered difficult**



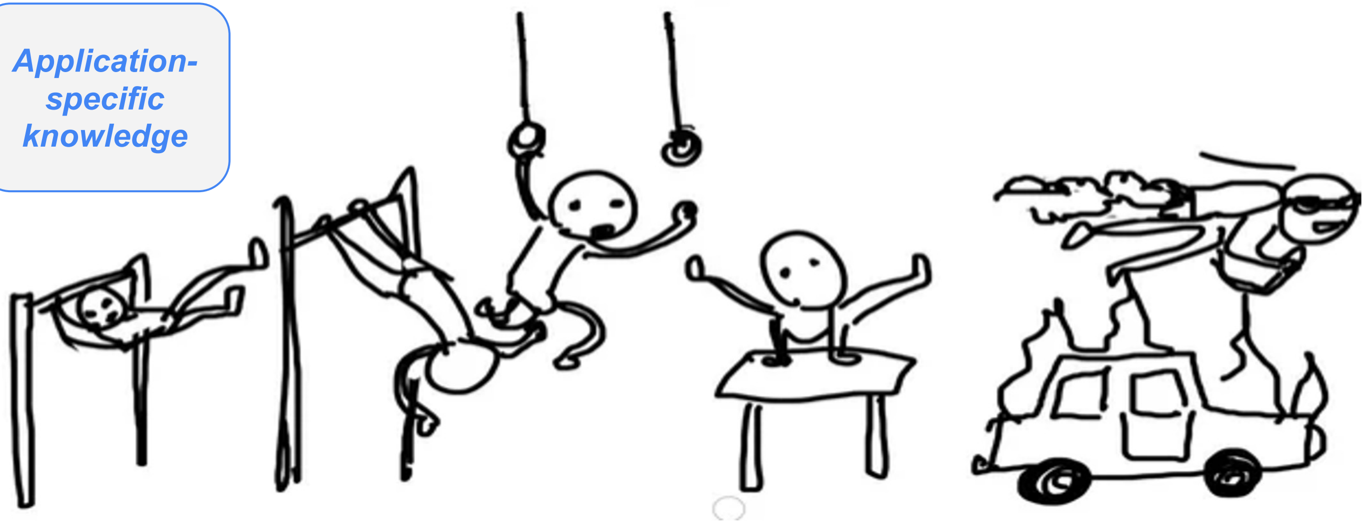
Why data-only attacks are **considered difficult**

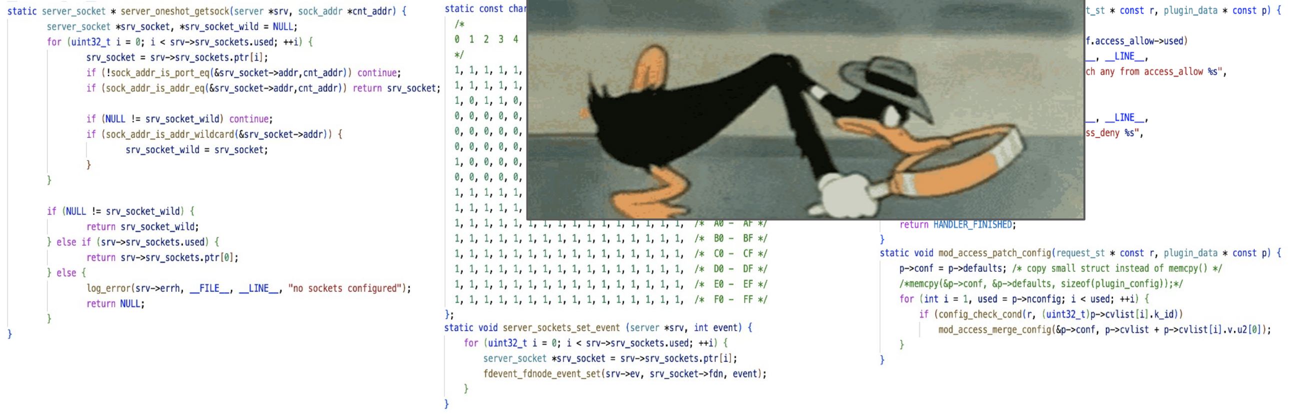
Application-specific knowledge



Why data-only attacks are **considered difficult**

Application-specific knowledge

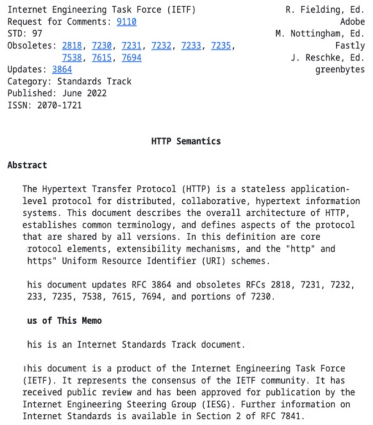




```
static server_socket * server_oneshot_getsock(server *srv, sock_addr *cnt_addr) {
        server_socket *srv_socket, *srv_socket_wild = NULL;
        for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
                srv_socket = srv->srv_sockets.ptr[i];
                if (!sock_addr_is_port_eq(&srv_socket->addr,cnt_addr)) continue;
                if (sock_addr_is_addr_eq(&srv_socket->addr,cnt_addr)) return srv_socket;

                if (NULL != srv_socket_wild) continue;
                if (sock_addr_is_addr_wildcard(&srv_socket->addr)) {
                        srv_socket_wild = srv_socket;
                }
        }

        if (NULL != srv_socket_wild) {
                return srv_socket_wild;
        } else if (srv->srv_sockets.used) {
                return srv->srv_sockets.ptr[0];
        } else {
                log_error(srv->errh, __FILE__, __LINE__, "no sockets configured");
                return NULL;
        }
}
```

```
static const char
  /*
  0  1  2  3  4
  */
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 0, 1, 1, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  A0 -  AF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  B0 -  BF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  C0 -  CF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  D0 -  DF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  E0 -  EF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  F0 -  FF */
};
static void server_sockets_set_event (server *srv, int event) {
    for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
        server_socket *srv_socket = srv->srv_sockets.ptr[i];
        fdevent_fdnode_event_set(srv->ev, srv_socket->fdn, event);
    }
}
```

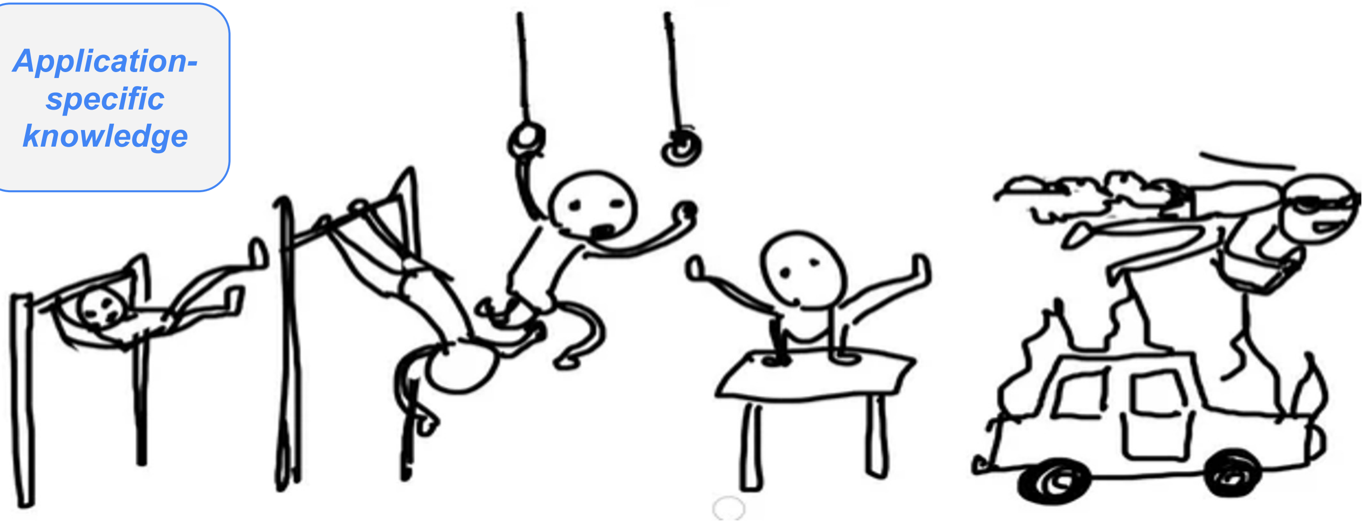
```
t_st * const r, plugin_data * const p) {

f.access_allow->used)
__, __LINE__,
ch any from access_allow %s",


__, __LINE__,
ss_deny %s",

    return HANDLER_FINISHED;
}
static void mod_access_patch_config(request_st * const r, plugin_data * const p) {
    p->conf = p->defaults; /* copy small struct instead of memcpy() */
    /*memcpy(&p->conf, &p->defaults, sizeof(plugin_config));*/
    for (int i = 1, used = p->nconfig; i < used; ++i) {
        if (config_check_cond(r, (uint32_t)p->cvlist[i].k_id))
            mod_access_merge_config(&p->conf, p->cvlist + p->cvlist[i].v.u2[0]);
    }
}
```

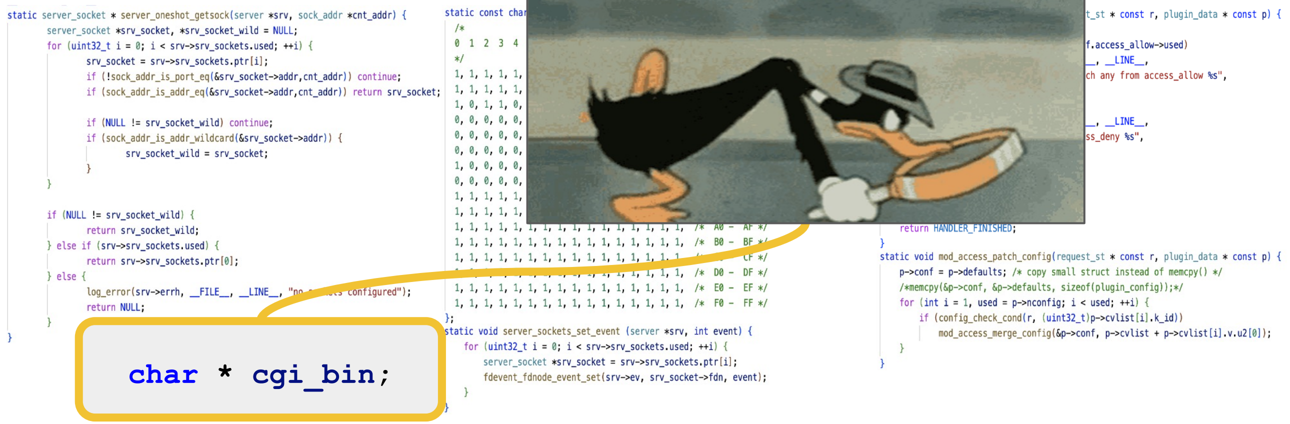


```
Internet Engineering Task Force (IETF)                    R. Fielding, Ed.
Request for Comments: 9110                                           Adobe
STD: 97                                                M. Nottingham, Ed.
Obsoletes: 2818, 7230, 7231, 7232, 7233, 7235,                      Fastly
           7538, 7615, 7694                                J. Reschke, Ed.
Updates: 3864                                                   greenbytes
Category: Standards Track
Published: June 2022
ISSN: 2070-1721


                              HTTP Semantics

Abstract

   The Hypertext Transfer Protocol (HTTP) is a stateless application-
   level protocol for distributed, collaborative, hypertext information
   systems. This document describes the overall architecture of HTTP,
   establishes common terminology, and defines aspects of the protocol
   that are shared by all versions. In this definition are core
    rotocol elements, extensibility mechanisms, and the "http" and
    https" Uniform Resource Identifier (URI) schemes.

    his document updates RFC 3864 and obsoletes RFCs 2818, 7231, 7232,
    233, 7235, 7538, 7615, 7694, and portions of 7230.

    us of This Memo

    his is an Internet Standards Track document.

   ıhis document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.
```

Why data-only attacks are **considered difficult**

Application-specific knowledge



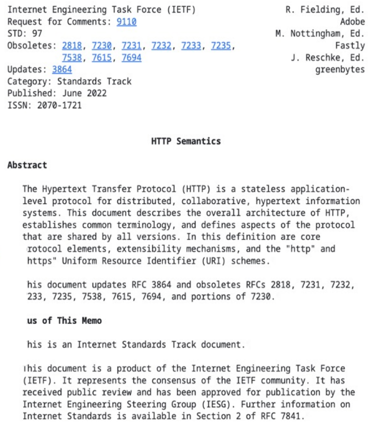


```
static server_socket * server_oneshot_getsock(server *srv, sock_addr *cnt_addr) {
        server_socket *srv_socket, *srv_socket_wild = NULL;
        for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
                srv_socket = srv->srv_sockets.ptr[i];
                if (!sock_addr_is_port_eq(&srv_socket->addr,cnt_addr)) continue;
                if (sock_addr_is_addr_eq(&srv_socket->addr,cnt_addr)) return srv_socket;

                if (NULL != srv_socket_wild) continue;
                if (sock_addr_is_addr_wildcard(&srv_socket->addr)) {
                        srv_socket_wild = srv_socket;
                }
        }

        if (NULL != srv_socket_wild) {
                return srv_socket_wild;
        } else if (srv->srv_sockets.used) {
                return srv->srv_sockets.ptr[0];
        } else {
                log_error(srv->errh, __FILE__, __LINE__, "no         configured");
                return NULL;
        }
}
```

```
static const char
  /*
  0  1  2  3  4
  */
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 0, 1, 1, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  A0 -  AF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  B0 -  BF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,          CF */
                                1, 1, 1, 1, 1, 1,  /*  D0 -  DF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  E0 -  EF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  F0 -  FF */
};
static void server_sockets_set_event (server *srv, int event) {
    for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
        server_socket *srv_socket = srv->srv_sockets.ptr[i];
        fdevent_fdnode_event_set(srv->ev, srv_socket->fdn, event);
    }
}
```

```
t_st * const r, plugin_data * const p) {

f.access_allow->used)
__, __LINE__,
ch any from access_allow %s",


__, __LINE__,
ss_deny %s",





    return HANDLER_FINISHED;
}
static void mod_access_patch_config(request_st * const r, plugin_data * const p) {
    p->conf = p->defaults; /* copy small struct instead of memcpy() */
    /*memcpy(&p->conf, &p->defaults, sizeof(plugin_config));*/
    for (int i = 1, used = p->nconfig; i < used; ++i) {
        if (config_check_cond(r, (uint32_t)p->cvlist[i].k_id))
            mod_access_merge_config(&p->conf, p->cvlist + p->cvlist[i].v.u2[0]);
    }
}
```

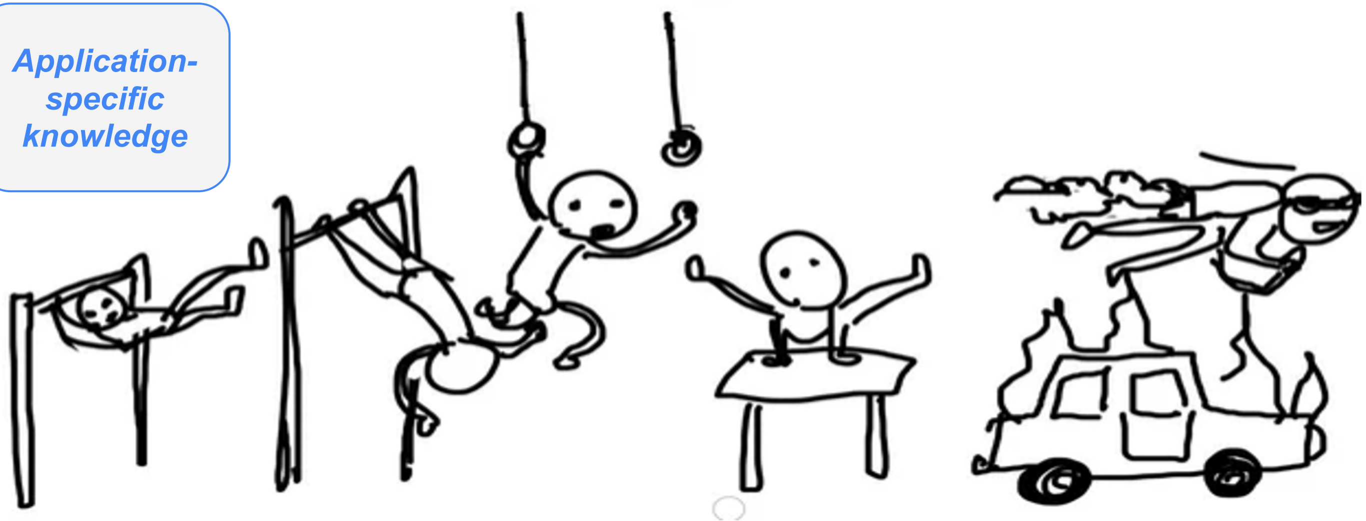
```
char * cgi_bin;
```



```
Internet Engineering Task Force (IETF)                    R. Fielding, Ed.
Request for Comments: 9110                                           Adobe
STD: 97                                                M. Nottingham, Ed.
Obsoletes: 2818, 7230, 7231, 7232, 7233, 7235,                      Fastly
           7538, 7615, 7694                                J. Reschke, Ed.
Updates: 3864                                                   greenbytes
Category: Standards Track
Published: June 2022
ISSN: 2070-1721


                             HTTP Semantics

Abstract

   The Hypertext Transfer Protocol (HTTP) is a stateless application-
   level protocol for distributed, collaborative, hypertext information
   systems. This document describes the overall architecture of HTTP,
   establishes common terminology, and defines aspects of the protocol
   that are shared by all versions. In this definition are core
    rotocol elements, extensibility mechanisms, and the "http" and
    https" Uniform Resource Identifier (URI) schemes.

    his document updates RFC 3864 and obsoletes RFCs 2818, 7231, 7232,
    233, 7235, 7538, 7615, 7694, and portions of 7230.

    us of This Memo

    his is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.
```

Why data-only attacks are **considered difficult**

Application-specific knowledge



```
static server_socket * server_oneshot_getsock(server *srv, sock_addr *cnt_addr) {
        server_socket *srv_socket, *srv_socket_wild = NULL;
        for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
                srv_socket = srv->srv_sockets.ptr[i];
                if (!sock_addr_is_port_eq(&srv_socket->addr,cnt_addr)) continue;
                if (sock_addr_is_addr_eq(&srv_socket->addr,cnt_addr)) return srv_socket;

        }

        if (NULL != srv_socket_wild) {
                return srv_socket_wild;
        } else if (srv->srv_sockets.used) {
                return srv->srv_sockets.ptr[0];
        } else {
                log_error(srv->errh, __FILE__, __LINE__, "no                configured");
                return NULL;
        }
}
```

HTTP PUT ...

```
static const char
  /*
  0  1  2  3  4
  */
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 0, 1, 1, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,                                                /*  A0 -  AF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  B0 -  BF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,           CF */
                                       1, 1, 1, 1,  /*  D0 -  DF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  E0 -  EF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  F0 -  FF */
};
static void server_sockets_set_event (server *srv, int event) {
    for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
        server_socket *srv_socket = srv->srv_sockets.ptr[i];
        fdevent_fdnode_event_set(srv->ev, srv_socket->fdn, event);
    }
}
```

```
t_st * const r, plugin_data * const p) {

f.access_allow->used)
_, __LINE__,
ch any from access_allow %s

_, __LINE__,
ss_deny %s",

    return HANDLER_FINISHED;
}
static void mod_access_patch_config(request_st * const r, plugin_data * const p) {
    p->conf = p->defaults; /* copy small struct instead of memcpy() */
    /*memcpy(&p->conf, &p->defaults, sizeof(plugin_config));*/
    for (int i = 1, used = p->nconfig; i < used; ++i) {
        if (config_check_cond(r, (uint32_t)p->cvlist[i].k_id))
                  onf, p->cvlist + p->cvlist[i].v.u2[0]);
```

HTTP HEAD ...

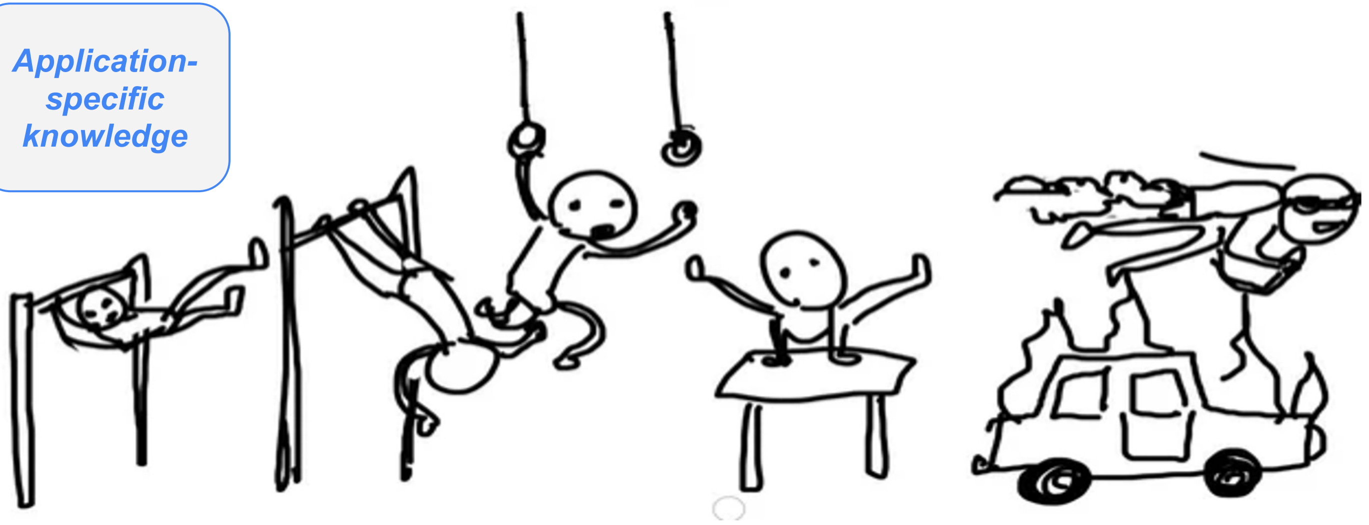
```
Internet Engineering Task Force (IETF)                 R. Fielding, Ed.
Request for Comments: 9110                                        Adobe
STD: 97                                            M. Nottingham, Ed.
                                   35,                           Fastly
                                                      J. Reschke, Ed.
                                                            greenbytes

                            HTTP Semantics

Abstract

   The Hypertext Transfer Protocol (HTTP) is a stateless application-
   level protocol for distributed, collaborative, hypertext information
   systems. This document describes the overall architecture of HTTP,
   establishes common terminology, and defines aspects of the protocol
   that are shared by all versions. In this definition are core
    rotocol elements, extensibility mechanisms, and the "http" and
    https" Uniform Resource Identifier (URI) schemes.

    his document updates RFC 3864 and obsoletes RFCs 2818, 7231, 7232,
    233, 7235, 7538, 7615, 7694, and portions of 7230.

    us of This Memo

    his is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.
```

HTTP GET ...

char * cgi_bin;

Why data-only attacks are **considered difficult**

Application-specific knowledge



```
static server_socket * server_oneshot_getsock(server *srv, sock_addr *cnt_addr) {
        server_socket *srv_socket, *srv_socket_wild = NULL;
        for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
                srv_socket = srv->srv_sockets.ptr[i];
                if (!sock_addr_is_port_eq(&srv_socket->addr,cnt_addr)) continue;
                if (sock_addr_is_addr_eq(&srv_socket->addr,cnt_addr)) return srv_socket;
```

```
HTTP PUT ...
```

```
        }

        if (NULL != srv_socket_wild) {
                return srv_socket_wild;
        } else if (srv->srv_sockets.used) {
                return srv->srv_sockets.ptr[0];
        } else {
                log_error(srv->errh, __FILE__, __LINE__, "no [illegible] configured");
                return NULL;
        }
}
```

```
char * cgi_bin;
```

```
static const char
  /*
  0  1  2  3  4
  */
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 0, 1, 1, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 0, 0, 0, 0,
  0, 0, 0, 0, 0,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  A0 -  AF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  B0 -  BF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  C0 -  CF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  D0 -  DF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  E0 -  EF */
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  /*  F0 -  FF */
};
static void server_sockets_set_event (server *srv, int event) {
    for (uint32_t i = 0; i < srv->srv_sockets.used; ++i) {
        server_socket *srv_socket = srv->srv_sockets.ptr[i];
        fdevent_fdnode_event_set(srv->ev, srv_socket->fdn, event);
    }
}
```

```
HTTP HEAD ...
```

```
t_st * const r, plugin_data * const p) {

f.access_allow->used)
_, __LINE__,
ch any from access_allow %s

_, __LINE__,
ss_deny %s",

    return HANDLER_FINISHED;
}
static void mod_access_patch_config(request_st * const r, plugin_data * const p) {
    p->conf = p->defaults; /* copy small struct instead of memc
    /*mem  (&p->conf, &p->defaults, sizeof(plugin_config));*/
    for (int i = 1, used   config; i < used; ++i) {
        if (config_check_cond(r, (uint32_t)p->cvlist[i].k_id)
                                onf, p->cvlist + p->
```

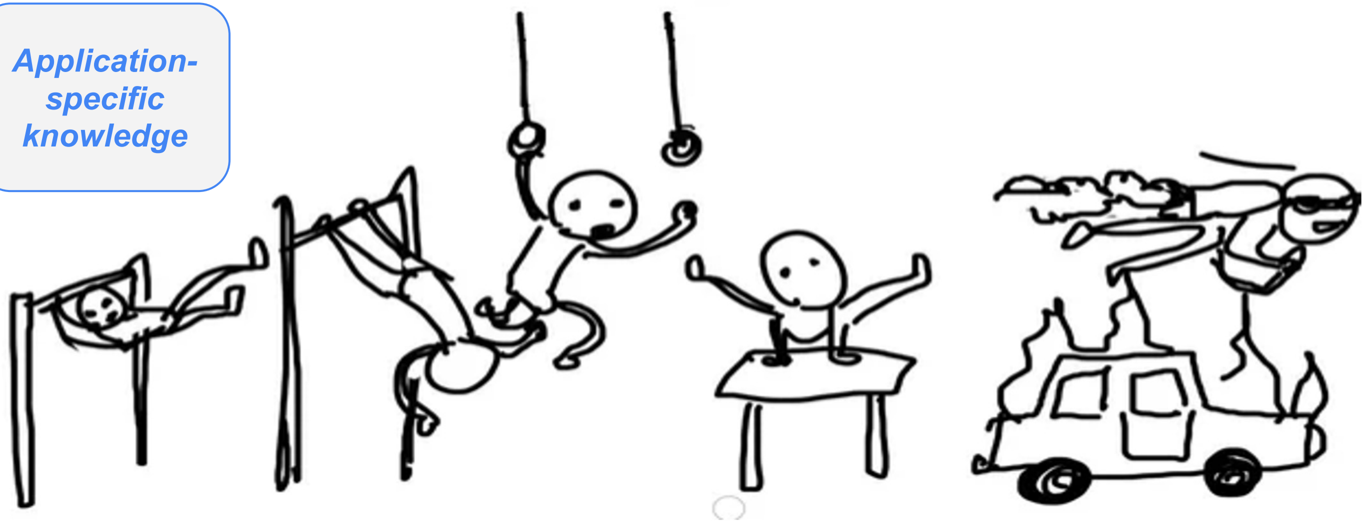
```
HTTP GET ...
```

```
HTTP POST /sh ...
```

```
Internet Engineering Task Force (IETF)                 R. Fielding, Ed.
Request for Comments: 9110                                        Adobe
STD: 97                                             M. Nottingham, Ed.
                                        35,                      Fastly
                                                      J. Reschke, Ed.
                                                             greenbytes

                              HTTP Semantics

Abstract

   The Hypertext Transfer Protocol (HTTP) is a stateless application-
   level protocol for distributed, collaborative, hypertext information
   systems. This document describes the overall architecture of HTTP,
   establishes common terminology, and defines aspects of the protocol
   that are shared by all versions. In this definition are core
   rotocol elements, extensibility mechanisms, and the "http" and
   https" Uniform Resource Identifier (URI) schemes.

                                          2818, 7231, 7232,
                                          0.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.
```

Why data-only attacks are **considered difficult**

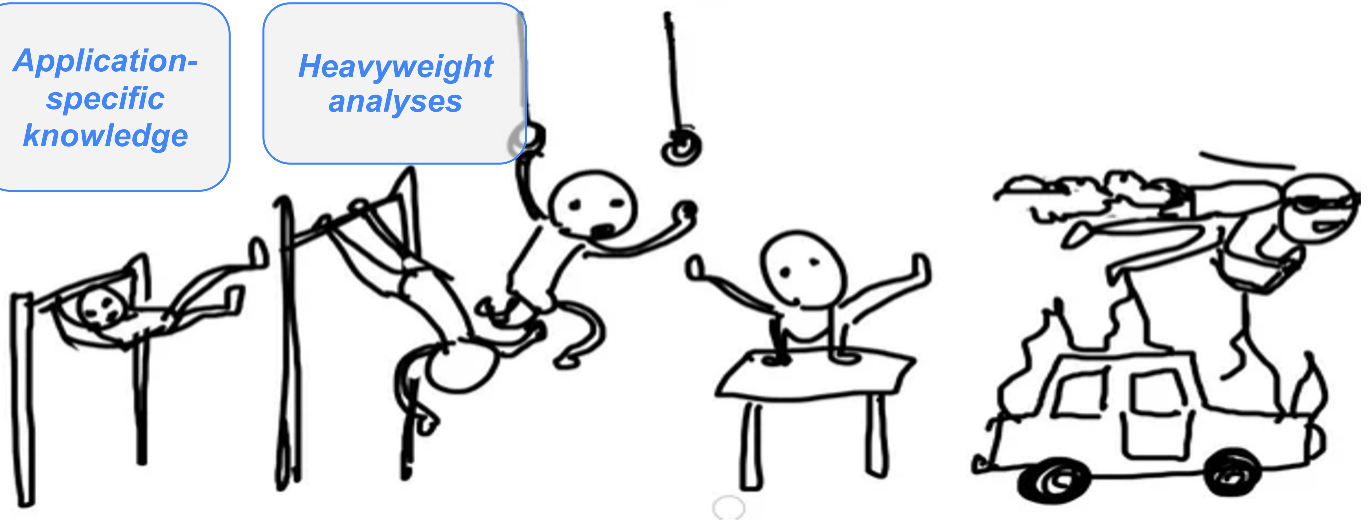
Application-specific knowledge



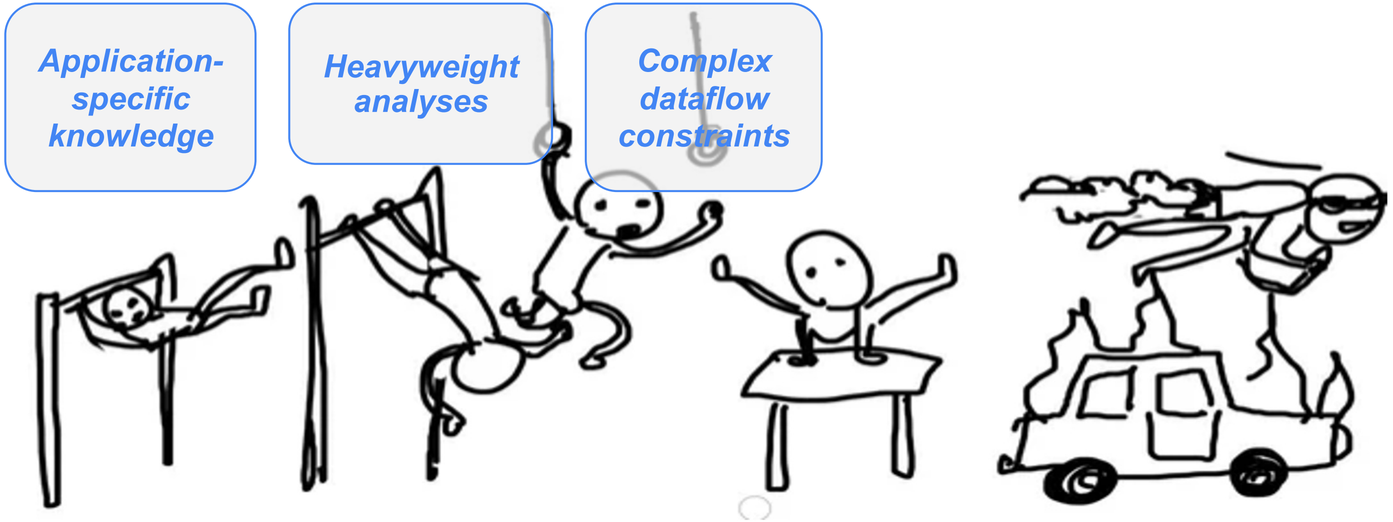
Why data-only attacks are **considered difficult**

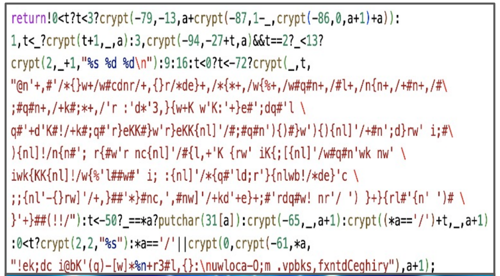
Application-specific knowledge

Heavyweight analyses



Why data-only attacks are **considered difficult**


Application-specific knowledge
Heavyweight analyses
Complex dataflow constraints



```
return!0<t?t<3?crypt(-79,-13,a+crypt(-87,1-_,crypt(-86,0,a+1)+a)):
1,t<_?crypt(t+1,_,a):3,crypt(-94,-27+t,a)&&t==2?_<13?
crypt(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?crypt(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;#\
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/"):t<-50?_==*a?putchar(31[a]):crypt(-65,_,a+1):crypt((*a=='/')+t,_,a+1)
:0<t?crypt(2,2,"%s"):*a=='/'||crypt(0,crypt(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-0;m .vpbks,fxntdCeghiry"),a+1);
```

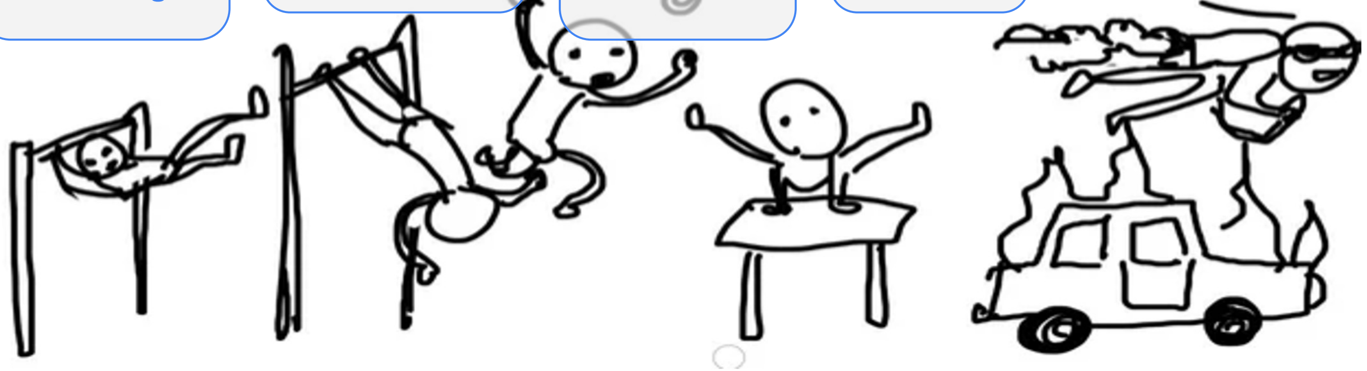
Why data-only attacks are **considered difficult**

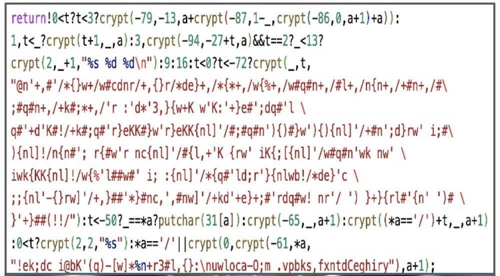
Application-specific knowledge

Heavyweight analyses

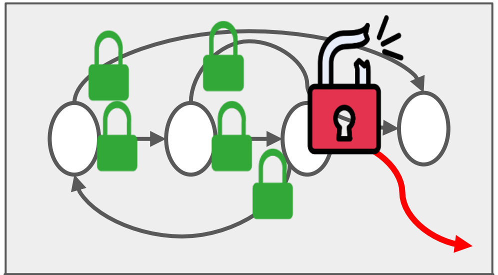
Complex dataflow constraints

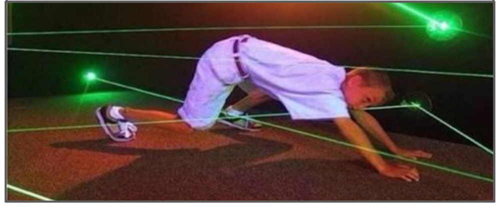
Bypass defenses





```
return!0<t?t<3?crypt(-79,-13,a+crypt(-87,1-_,crypt(-86,0,a+1)+a)):
1,t<_?crypt(t+1,_,a):3,crypt(-94,-27+t,a)&&t==2?_<13?
crypt(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?crypt(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;#\
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/"):t<-50?_==*a?putchar(31[a]):crypt(-65,_,a+1):crypt((*a=='/')+t,_,a+1)
:0<t?crypt(2,2,"%s"):*a=='/'||crypt(0,crypt(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-0;m .vpbks,fxntdCeghiry"),a+1);
```





Why data-only attacks are **considered difficult**

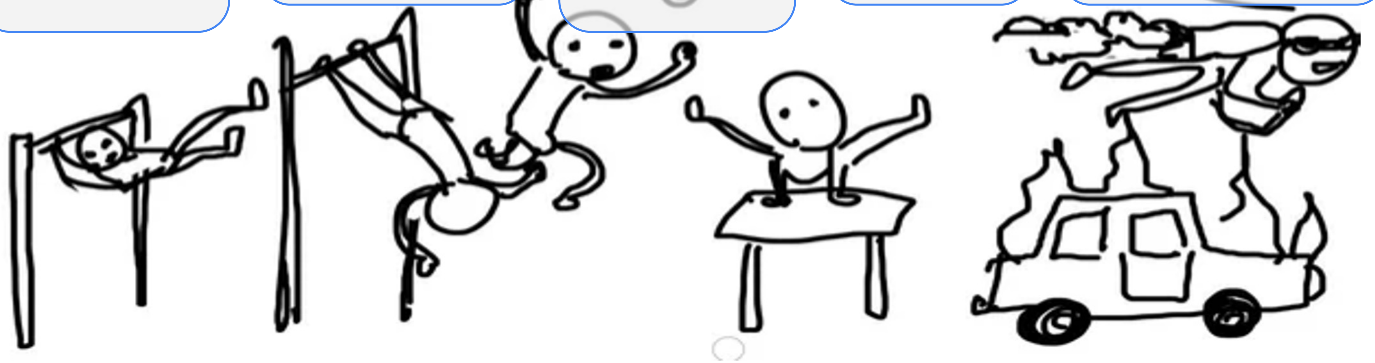
Application-specific knowledge

Heavyweight analyses

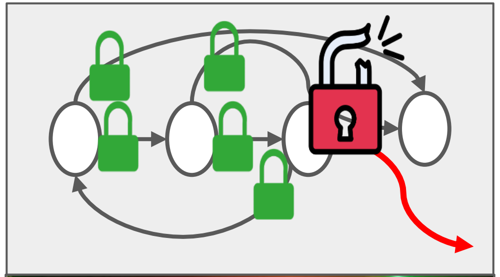
Complex dataflow constraints

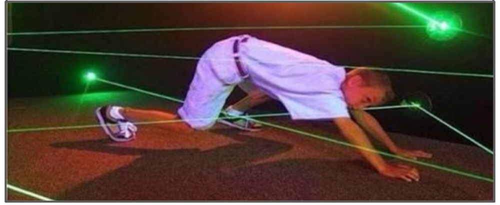
Bypass defenses

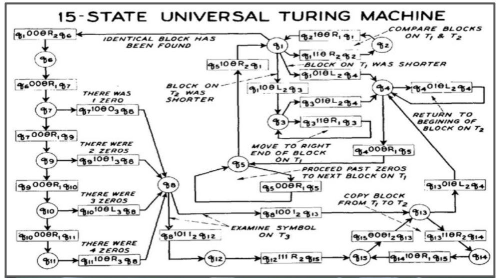
Turing-complete gadget set



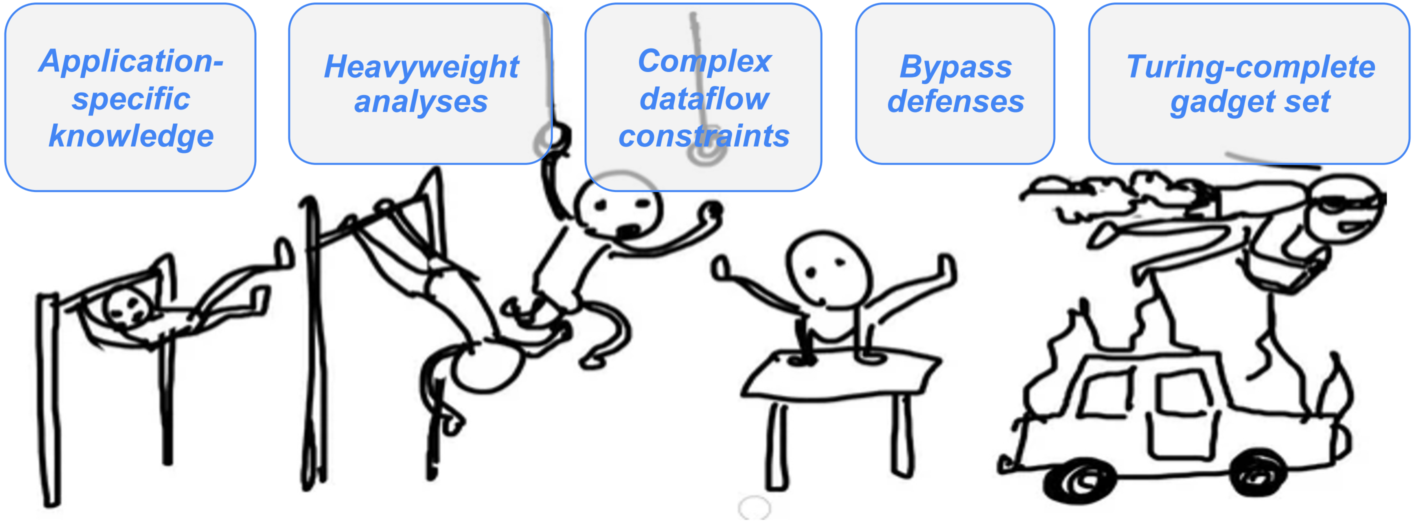
```
return!0<t?t<3?crypt(-79,-13,a+crypt(-87,1-_,crypt(-86,0,a+1)+a)):
1,t<_?crypt(t+1,_,a):3,crypt(-94,-27+t,a)&&t==2?_<13?
crypt(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?crypt(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;#\
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/"):t<-50?_==*a?putchar(31[a]):crypt(-65,_,a+1):crypt((*a=='/')+t,_,a+1)
:0<t?crypt(2,2,"%s"):*a=='/'||crypt(0,crypt(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);
```






15-STATE UNIVERSAL TURING MACHINE
IDENTICAL BLOCK HAS BEEN FOUND
COMPARE BLOCKS ON T1 & T2
BLOCK ON T1 WAS SHORTER
THERE WAS 1 ZERO
BLOCK ON T2 WAS SHORTER
THERE WERE 2 ZEROS
RETURN TO BEGINING OF BLOCK ON T2
MOVE TO RIGHT END OF BLOCK ON T1
PROCEED PAST ZEROS TO NEXT BLOCK ON T1
THERE WERE 3 ZEROS
COPY BLOCK FROM T1 TO T2
EXAMINE SYMBOL ON T3
THERE WERE 4 ZEROS

Why data-only attacks are **considered difficult**


Application-specific knowledge
Heavyweight analyses
Complex dataflow constraints
Bypass defenses
Turing-complete gadget set

How data-only attacks **can be**

Why data-only attacks are **considered difficult**

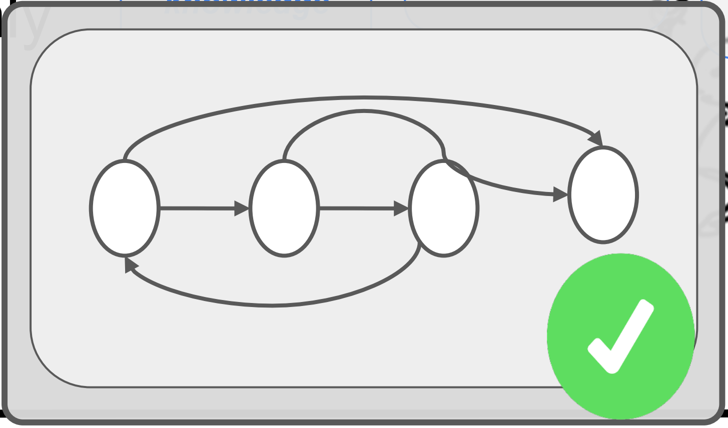
Application-specific knowledge

Heavyweight analyses

Complex dataflow constraints

Bypass defenses

Turing-complete gadget set



How data-only attacks **can be**

Victim executes its intended code

Why data-only attacks are **considered difficult**

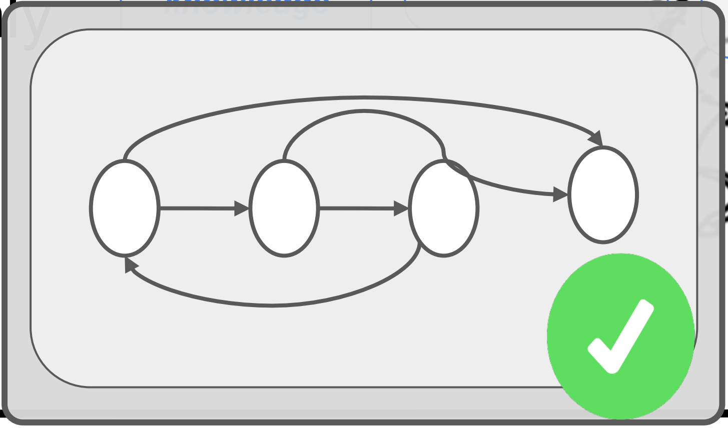
Application-specific knowledge

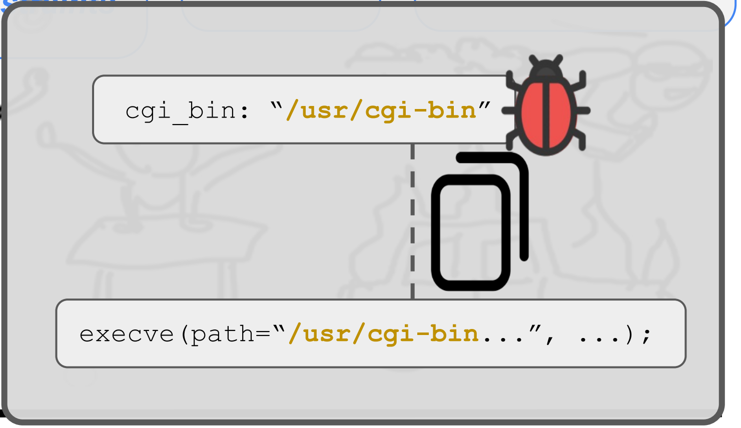
Heavyweight analyses

Complex dataflow constraints

Bypass defenses

Turing-complete gadget set





```
cgi_bin: "/usr/cgi-bin"
```

```
execve(path="/usr/cgi-bin...", ...);
```

How data-only attacks **can be**

Victim executes its intended code

Victim passes attacker data to a syscall verbatim

Why data-only attacks are **considered difficult**

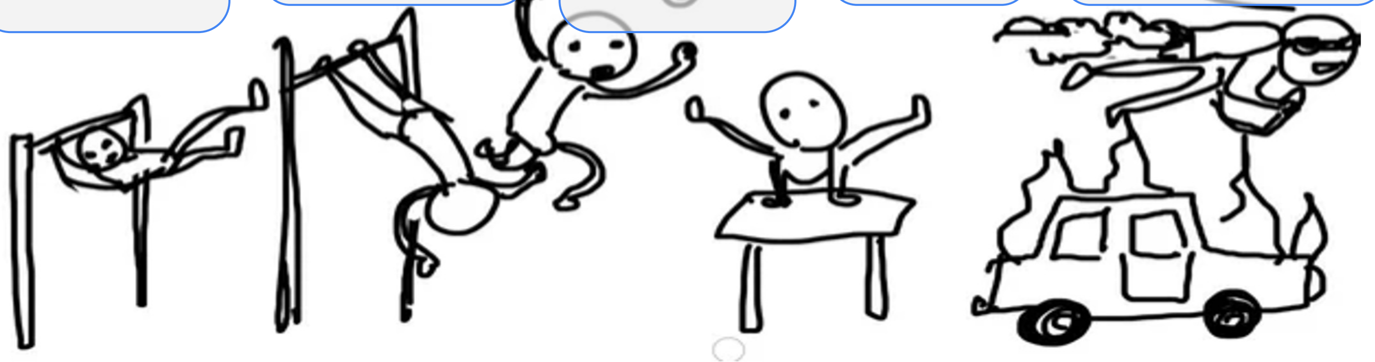
Application-specific knowledge

Heavyweight analyses

Complex dataflow constraints

Bypass defenses

Turing-complete gadget set



How data-only attacks **can be**

Victim executes its intended code

Victim passes attacker data to a syscall verbatim

Einstein: “As simple as possible, but not simpler”

Einstein: “As simple as possible, but not simpler”

❓ ***Which*** *data to overwrite?*

Einstein: “As simple as possible, but not simpler”

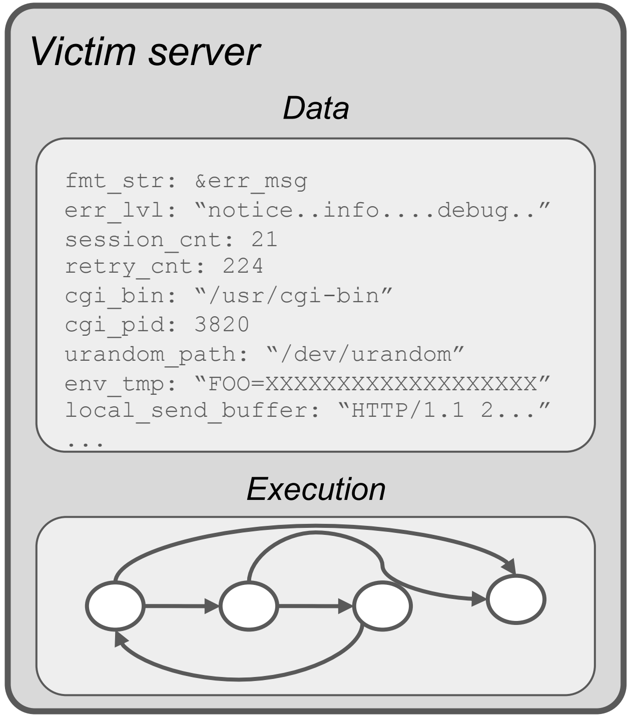
❓ ***Which** data to overwrite?*

❓ ***What** to overwrite it with?*

Einstein: “As simple as possible, but not simpler”

? ***Which** data to overwrite?*

? ***What** to overwrite it with?*

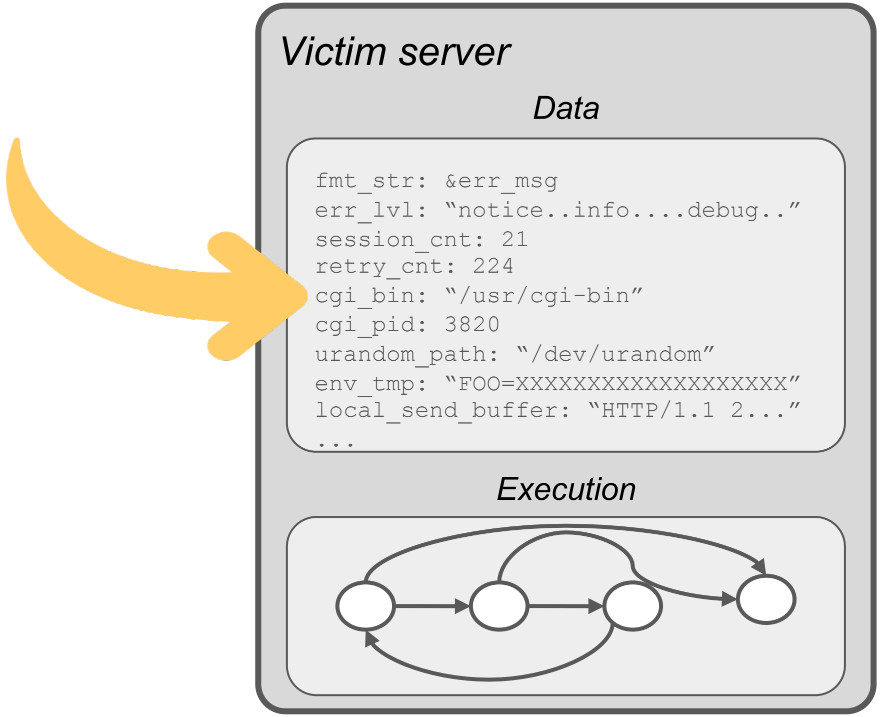


```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

Einstein: “As simple as possible, but not simpler”

***Which** data to overwrite?*

***What** to overwrite it with?*



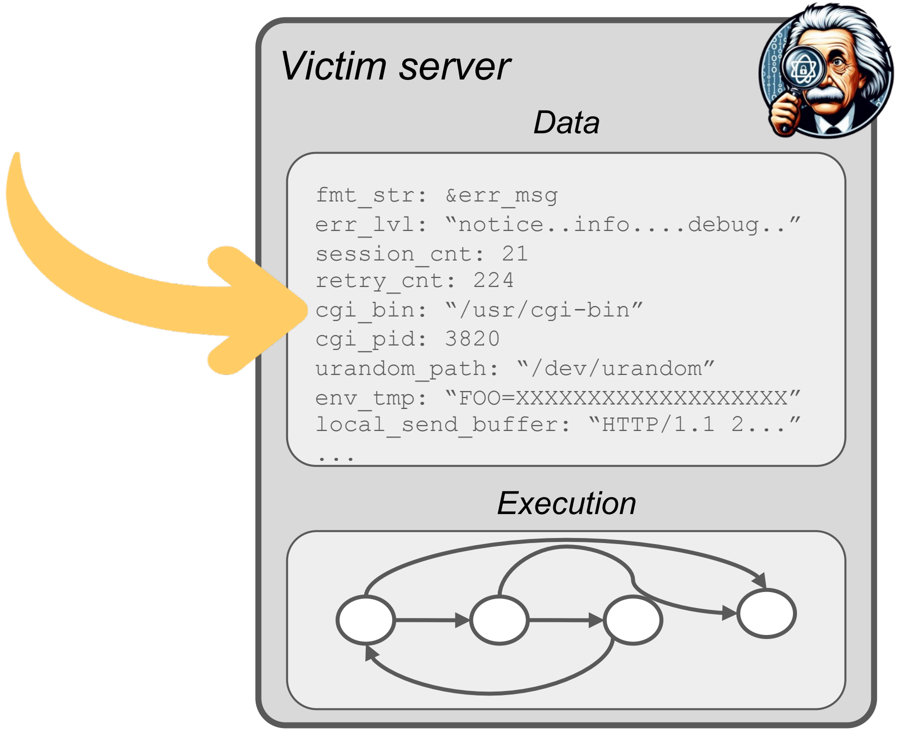
```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

Einstein: “As simple as possible, but not simpler”

❓ ***Which*** *data to overwrite?*

💡 *Use* ***dynamic taint analysis***

❓ ***What*** *to overwrite it with?*



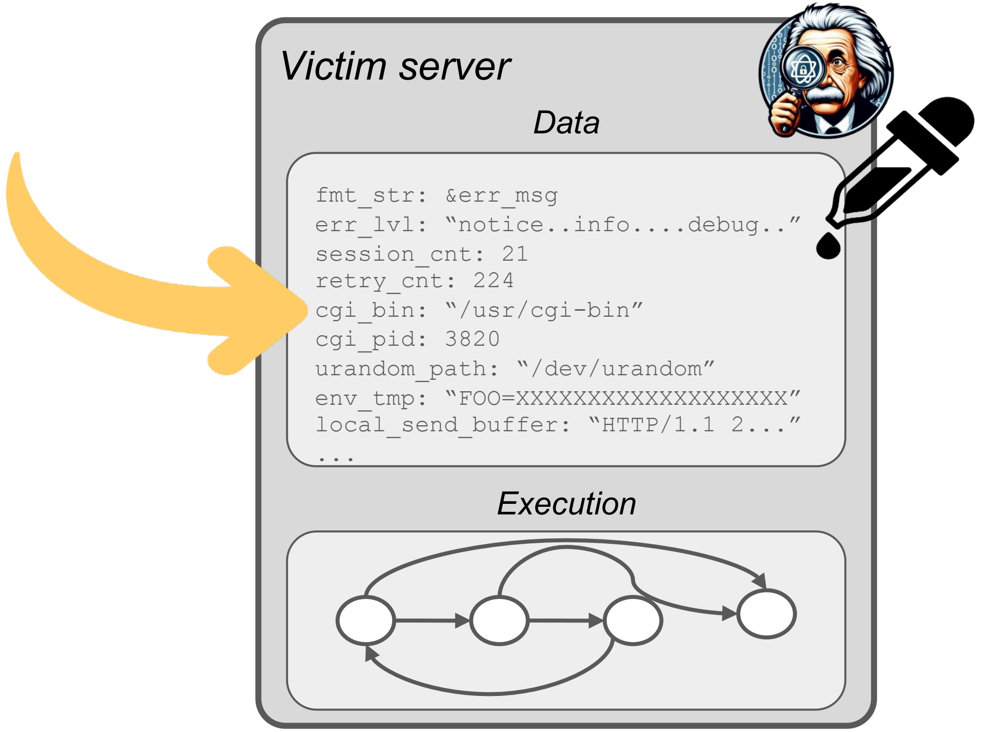
```
fmt_str: &err_msg
err_lvl: "notice..info....debug.."
session_cnt: 21
retry_cnt: 224
cgi_bin: "/usr/cgi-bin"
cgi_pid: 3820
urandom_path: "/dev/urandom"
env_tmp: "FOO=XXXXXXXXXXXXXXXXXX"
local_send_buffer: "HTTP/1.1 2..."
...
```

Einstein: “As simple as possible, but not simpler”

Which *data to overwrite?*

Use ***dynamic taint analysis***

What *to overwrite it with?*



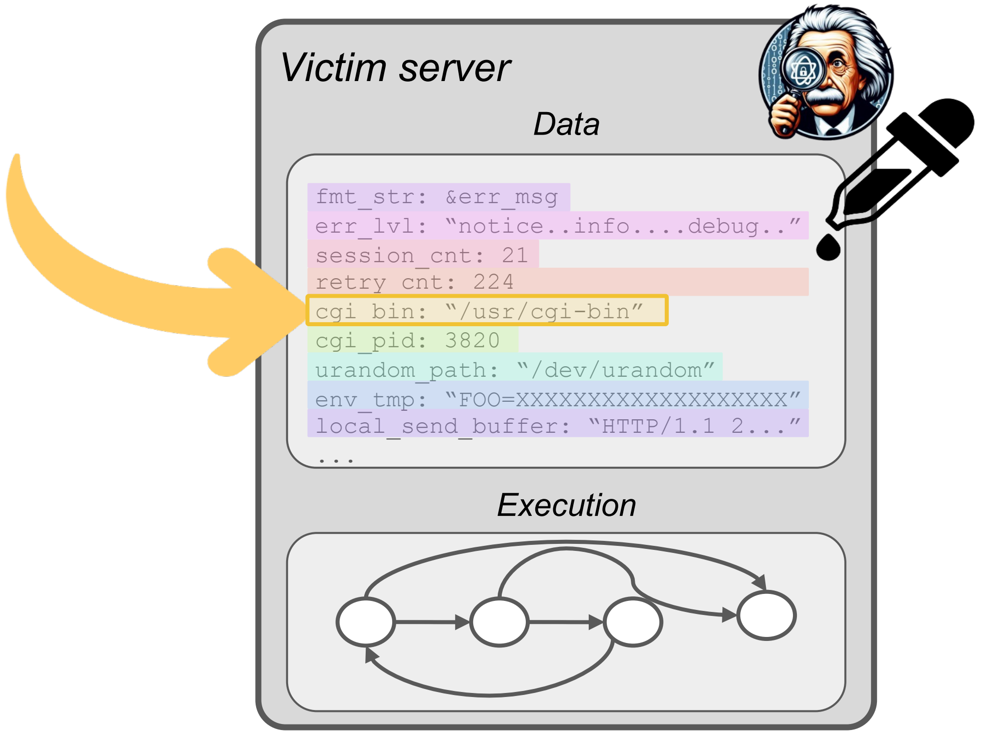
```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

Einstein: "As simple as possible, but not simpler"

***Which** data to overwrite?*

*Use **dynamic taint analysis***

***What** to overwrite it with?*



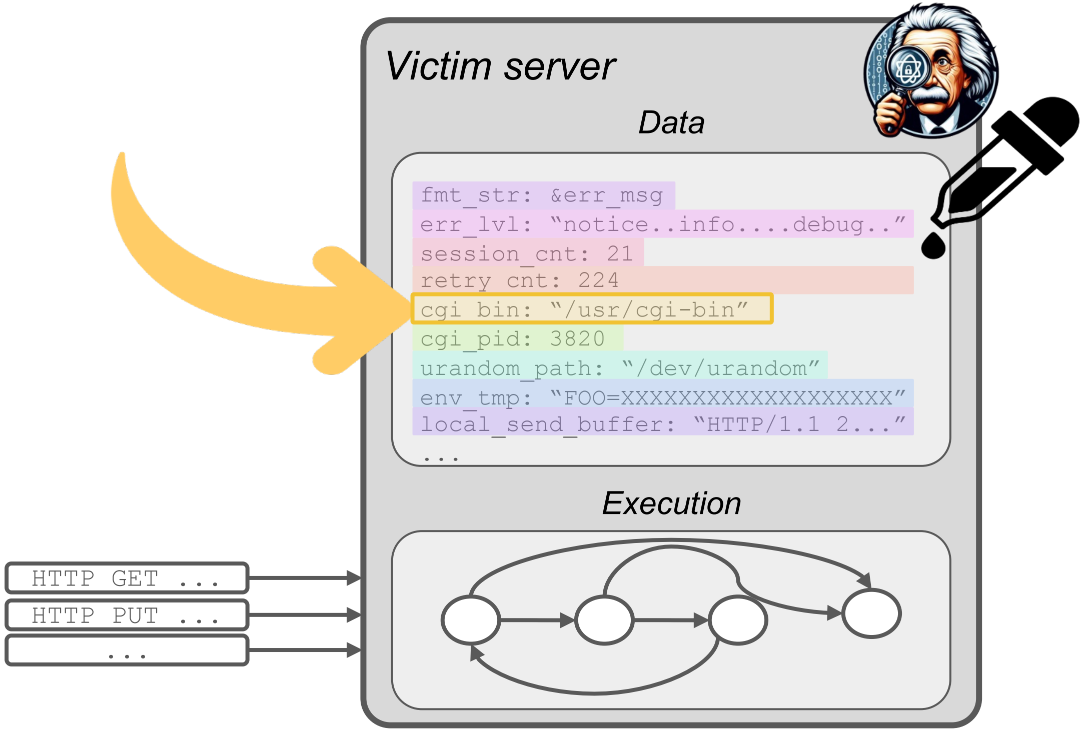
```
fmt_str: &err_msg
err_lvl: "notice..info....debug.."
session_cnt: 21
retry_cnt: 224
cgi_bin: "/usr/cgi-bin"
cgi_pid: 3820
urandom_path: "/dev/urandom"
env_tmp: "FOO=XXXXXXXXXXXXXXXXXX"
local_send_buffer: "HTTP/1.1 2..."
...
```

Einstein: “As simple as possible, but not simpler”

Which *data to overwrite?*

Use ***dynamic taint analysis***

What *to overwrite it with?*

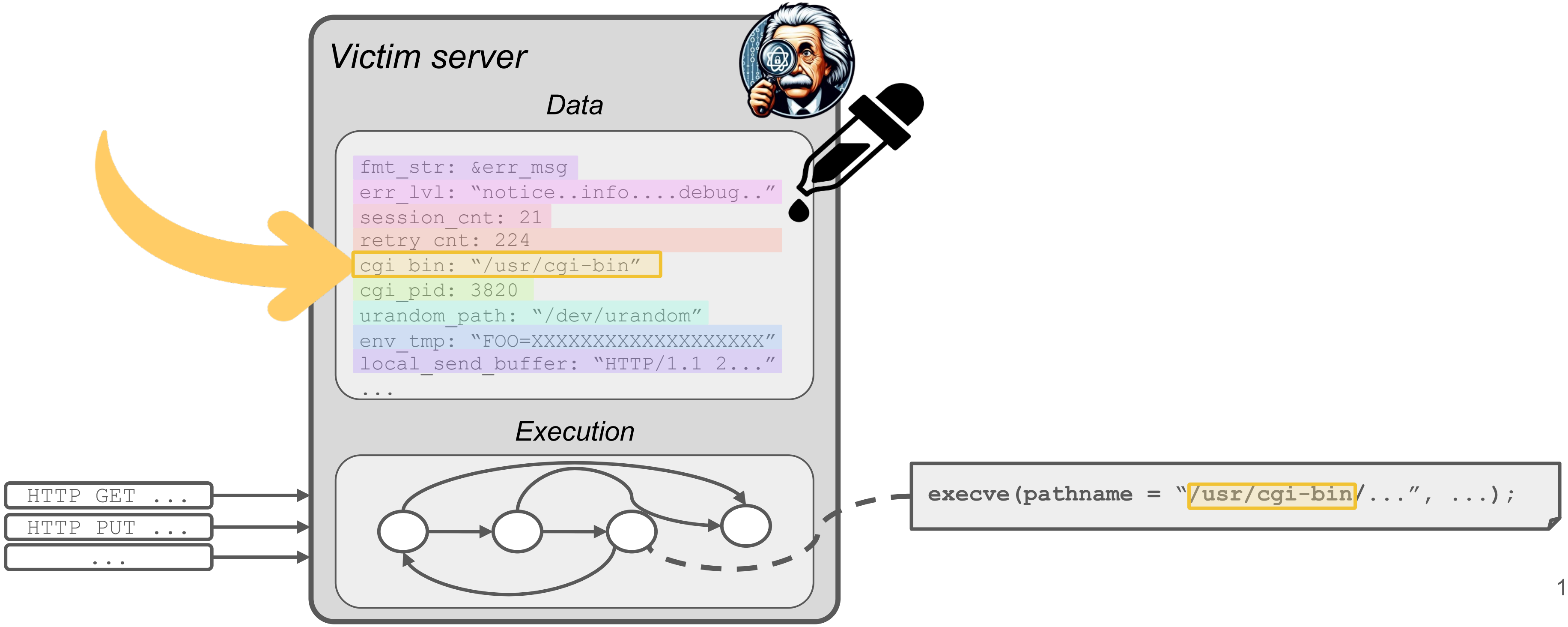


Einstein: “As simple as possible, but not simpler”

Which *data to overwrite?*

Use ***dynamic taint analysis***

What *to overwrite it with?*



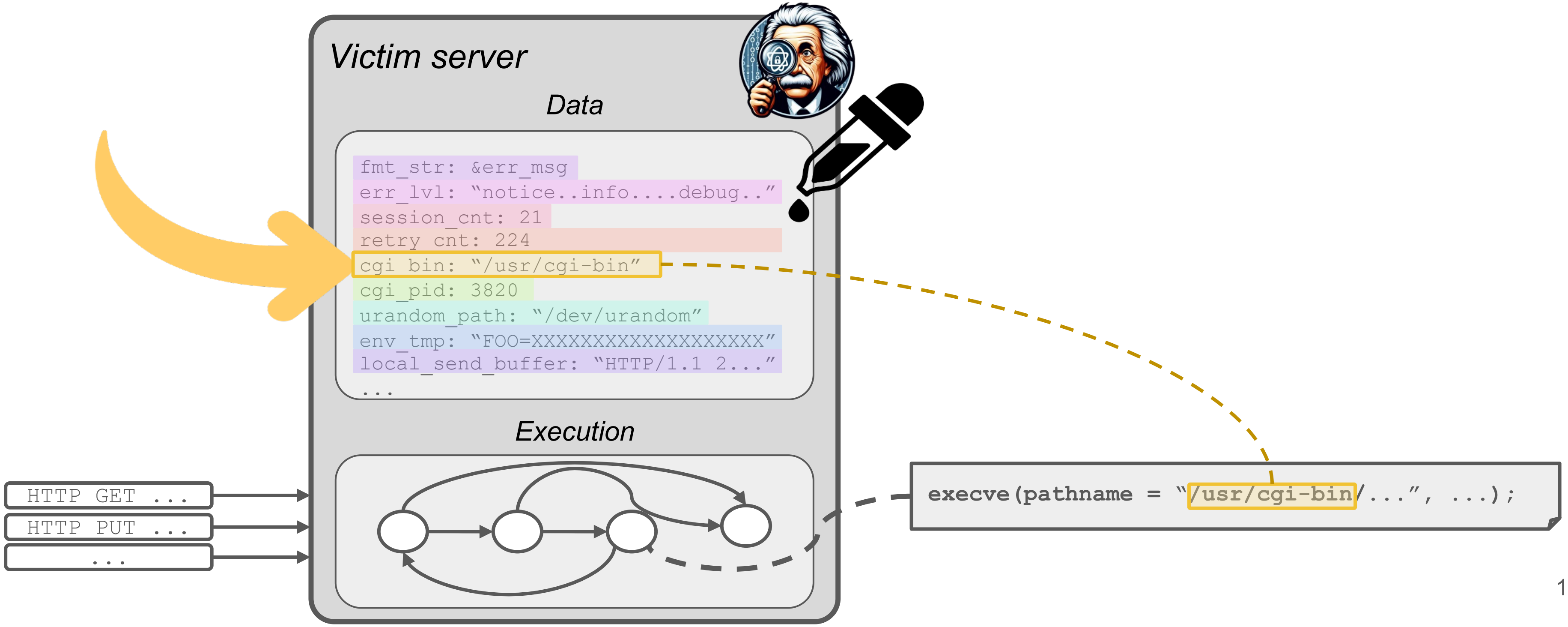
Victim server
Data
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
Execution
HTTP GET ...
HTTP PUT ...
...
execve(pathname = “/usr/cgi-bin/...”, ...);

Einstein: “As simple as possible, but not simpler”

Which *data to overwrite?*

Use ***dynamic taint analysis***

What *to overwrite it with?*



```
fmt_str: &err_msg
err_lvl: "notice..info....debug.."
session_cnt: 21
retry_cnt: 224
cgi_bin: "/usr/cgi-bin"
cgi_pid: 3820
urandom_path: "/dev/urandom"
env_tmp: "FOO=XXXXXXXXXXXXXXXXXX"
local_send_buffer: "HTTP/1.1 2..."
...
```

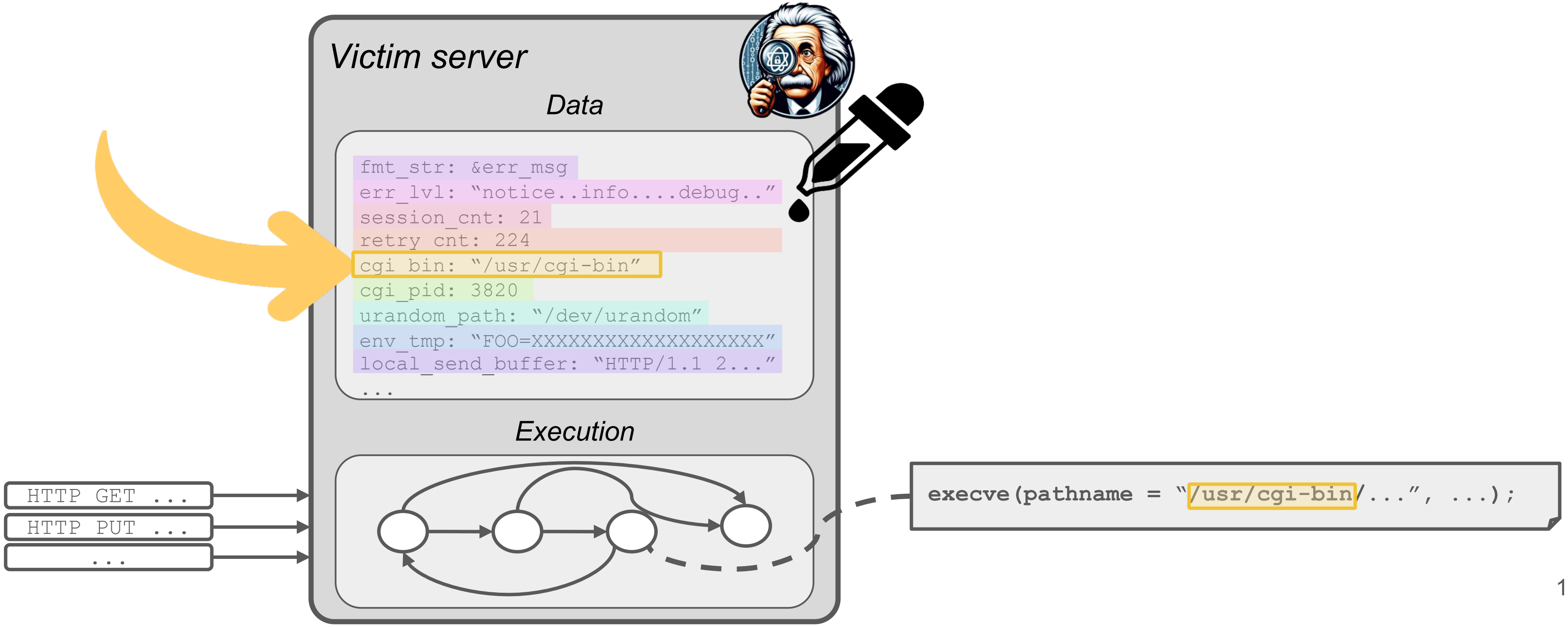
```
execve(pathname = "/usr/cgi-bin/...", ...);
```

Einstein: “As simple as possible, but not simpler”

***Which** data to overwrite?*

*Use **dynamic taint analysis***

***What** to overwrite it with?*



```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

```
execve(pathname = “/usr/cgi-bin/...”, ...);
```

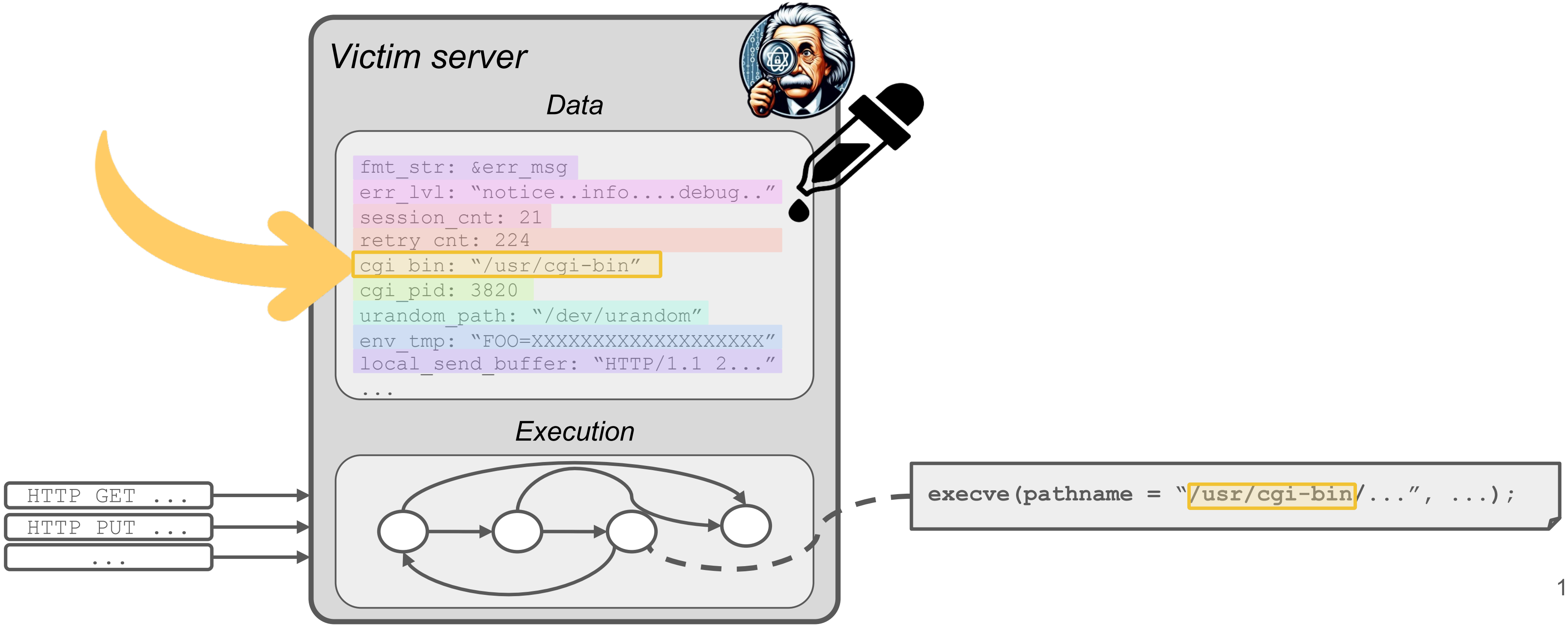
Einstein: “As simple as possible, but not simpler”

***Which** data to overwrite?*

*Use **dynamic taint analysis***

***What** to overwrite it with?*

*Exploit data copied **verbatim***



```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

```
execve(pathname = “/usr/cgi-bin/...”, ...);
```

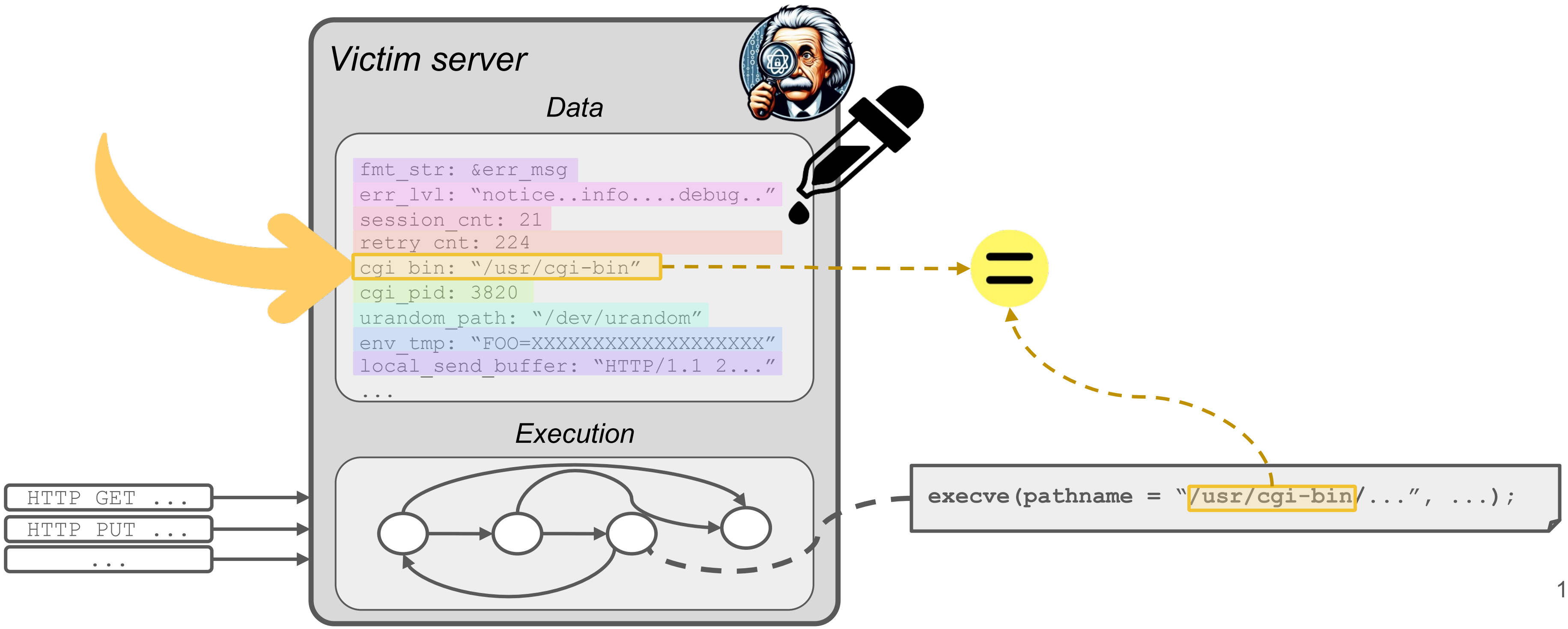
Einstein: “As simple as possible, but not simpler”

***Which** data to overwrite?*

*Use **dynamic taint analysis***

***What** to overwrite it with?*

*Exploit data copied **verbatim***



```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

```
execve(pathname = “/usr/cgi-bin/...”, ...);
```

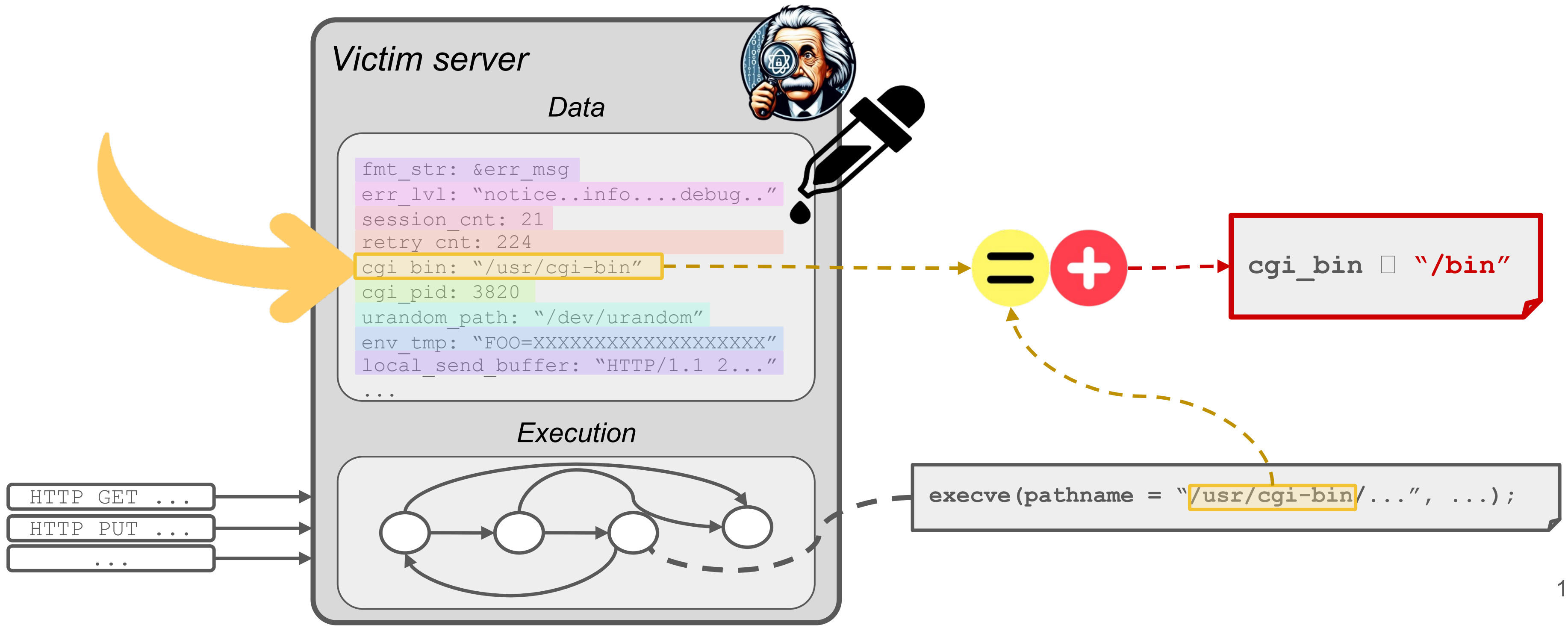
Einstein: “As simple as possible, but not simpler”

***Which** data to overwrite?*

*Use **dynamic taint analysis***

***What** to overwrite it with?*

*Exploit data copied **verbatim***



```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

```
cgi_bin □ “/bin”
```

```
execve(pathname = “/usr/cgi-bin/...”, ...);
```

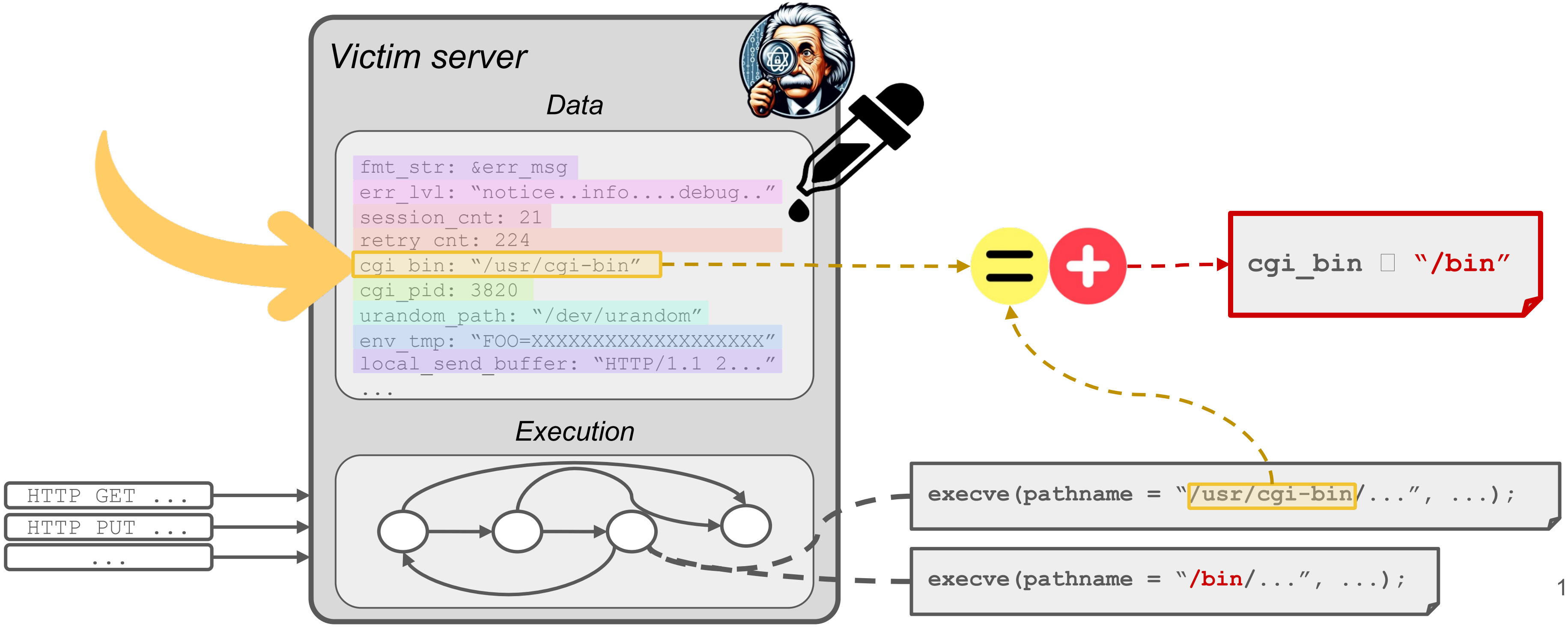
Einstein: “As simple as possible, but not simpler”

***Which** data to overwrite?*

*Use **dynamic taint analysis***

***What** to overwrite it with?*

*Exploit data copied **verbatim***



```
execve(pathname = "/usr/cgi-bin/...", ...);
execve(pathname = "/bin/...", ...);
```

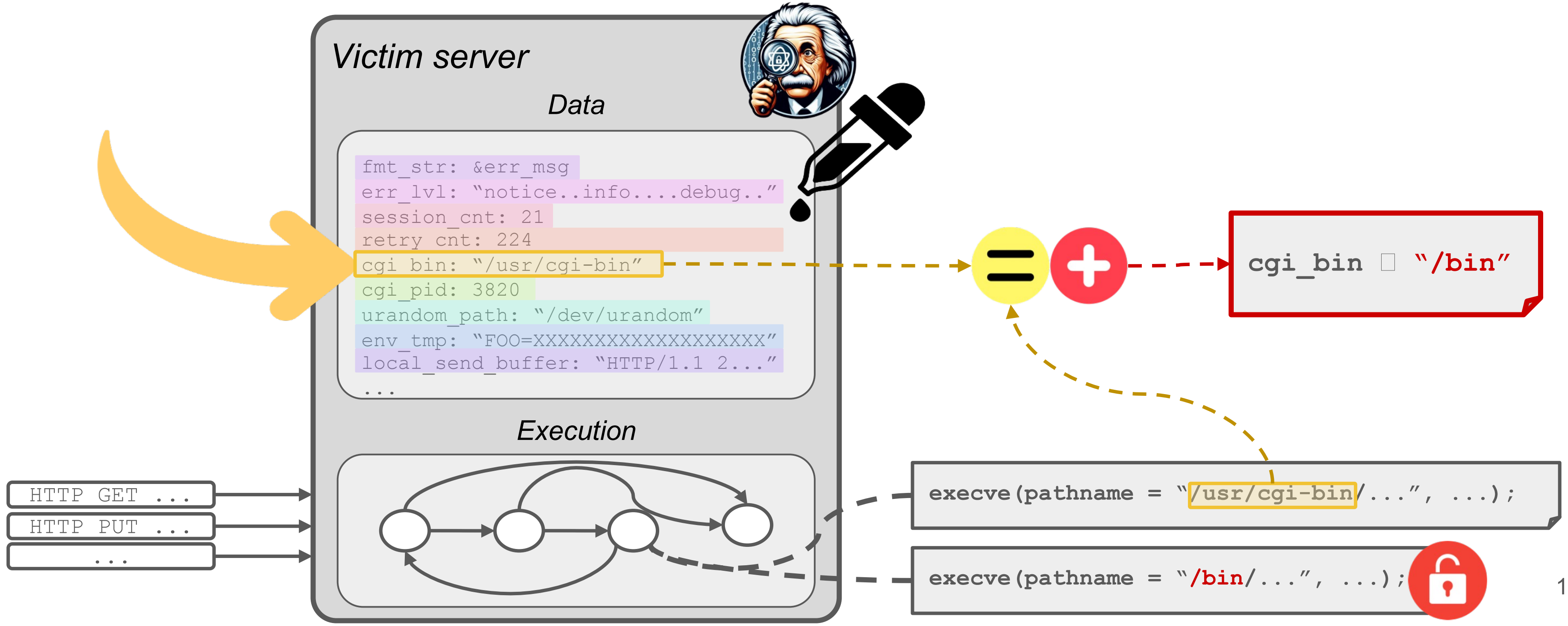
Einstein: “As simple as possible, but not simpler”

Which *data to overwrite?*

Use ***dynamic taint analysis***

What *to overwrite it with?*

Exploit data copied ***verbatim***



```
fmt_str: &err_msg
err_lvl: “notice..info....debug..”
session_cnt: 21
retry_cnt: 224
cgi_bin: “/usr/cgi-bin”
cgi_pid: 3820
urandom_path: “/dev/urandom”
env_tmp: “FOO=XXXXXXXXXXXXXXXXXX”
local_send_buffer: “HTTP/1.1 2...”
...
```

```
cgi_bin  “/bin”
```

```
execve(pathname = “/usr/cgi-bin/...”, ...);
```

```
execve(pathname = “/bin/...”, ...);
```

Evaluation: Setup

Evaluation: Setup

Target applications:

Evaluation: Setup

Target applications:

NGINX

APACHE HTTP SERVER

LIGHTTPD fly light.

PostgreSQL

redis

Evaluation: Setup

Target applications:

CVE-2022-23943
CVE-2022-41741
CVE-2007-4727
CVE-2023-36824
CVE-2021-32027
...

Evaluation: Setup

Target applications:

NGINX APACHE HTTP SERVER LIGHTTPD fly light. PostgreSQL redis

CVE-2022-23943
CVE-2022-41741
CVE-2007-4727
CVE-2023-36824
CVE-2021-32027
...

Workloads:

Evaluation: Setup

Target applications:

NGINX

APACHE HTTP SERVER

LIGHTTPD fly light.

PostgreSQL

redis

CVE-2022-23943
CVE-2022-41741
CVE-2007-4727
CVE-2023-36824
CVE-2021-32027
...

Workloads: *Each application's test suite*

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%)	7 (86%)	7 (86%)			

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%)	7 (86%)	7 (86%)			

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%)	7 (86%)	7 (86%)			

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) *`pathname`*	7 (86%) *`argv`*	7 (86%) *`envp`*			

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) `pathname`	7 (86%) `argv`	7 (86%) `envp`			

*After initialization, programs typically copy **strings** around **verbatim**.*

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	**Arg. 2**	**Arg. 3**	**Arg. 4**	**Arg. 5**	**Arg. 6**
`execve`	11	7 (86%) *pathname*	7 (86%) *argv*	7 (86%) *envp*			
`mmap`	787	55 (5%)	441 (8%)	55	16 (100%)	17	73
`mprotect`	144	97 (68%)	98 (27%)	1			
`mremap`	11	11	7	11	-		
`pwrite64`	1270	1217 (3%)	741 (47%)	399 (19%)	506 (36		
`pwritev`	10	10 (100%)	10 (10%)	-	10 (20%)		
`sendfile`	1	-	-	1	1		
`sendmmsg`	2	2	2	-	-		
`sendmsg`	12	5 (100%)	3 (100%)	-			
`sendto`	431	389 (7%)	410 (38%)	408 (6%)	-	-	-
`write`	3083	768 (74%)	2877 (91%)	1265 (10%)			
`writev`	754	244 (18%)	742 (76%)	-			

*After initialization, programs typically copy **strings** around **verbatim**.*

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) *pathname*	7 (86%) *argv*	7 (86%) *envp*			
`mmap`	787	55 (5%)	441 (8%)	55	16 (100%)	17	73
`mprotect`	144	97 (68%)	98 (27%)	1			
`mremap`	11	11	7	11	-		
`pwrite64`	1270	1217 (3%)	741 (47%)	399 (19%)	506 (36		
`pwritev`	10	10 (100%)	10 (10%)	-	10 (20%)		
`sendfile`	1	-	-	1	1		
`sendmmsg`	2	2	2	-	-		
`sendmsg`	12	5 (100%)	3 (100%)	-			
`sendto`	431	389 (7%)	410 (38%)	408 (6%)	-	-	-
`write`	3083	768 (74%)	2877 (91%)	1265 (10%)			
`writev`	754	244 (18%)	742 (76%)	-			

After initialization, programs typically copy ***strings*** *around* ***verbatim***.

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) *pathname*	7 (86%) *argv*	7 (86%) *envp*			
`mmap`	787	55 (5%)	441 (8%)	55	16 (100%)	17	73
`mprotect`	144	97 (68%)	98 (27%)	1			
`mremap`	11	11	7	11	-		
`pwrite64`	1270	1217 (3%)	741 (47%)	399 (19%)	506 (36		
`pwritev`	10	10 (100%)	10 (10%)	-	10 (20%)		
`sendfile`	1	-	-	1	1		
`sendmmsg`	2	2	2	-	-		
`sendmsg`	12	5 (100%)	3 (100%)	-			
`sendto`	431	389 (7%)	410 (38%)	408 (6%)	-		
`write`	3083	768 (74%)	2877 (91%)	1265 (10%)			
`writev`	754	244 (18%)	742 (76%)	-			

*After initialization, programs typically copy **strings** around **verbatim**.*

*Many types of syscalls offer many **primitives to an attacker**.*

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) *pathname*	7 (86%) *argv*	7 (86%) *envp*			
`mmap`	787	55 (5%)	441 (8%)	55	16 (100%)	17	73
`mprotect`	144	97 (68%)	98 (27%)	1			
`mremap`	11	11	7	11	-		
`pwrite64`	1270	1217 (3%)	741 (47%)	399 (19%)	506 (36		
`pwritev`	10	10 (100%)	10 (10%)	-	10 (20%)		
`sendfile`	1	-	-	1	1		
`sendmmsg`	2	2	2	-	-		
`sendmsg`	12	5 (100%)	3 (100%)	-			
`sendto`	431	389 (7%)	410 (38%)	408 (6%)	-		
`write`	3083	768 (74%)	2877 (91%)	1265 (10%)			
`writev`	754	244 (18%)	742 (76%)	-			

After initialization, programs typically copy ***strings*** *around* ***verbatim****.*

Many types of syscalls offer many ***primitives to an attacker****.*

Evaluation

Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) `pathname`	7 (86%) `argv`	7 (86%) `envp`			
`mmap`	787	55 (5%)	441 (8%)	55	16 (100%)	17	73
`mprotect`	144	97 (68%)	98 (27%)	1			
`mremap`	11	11	7	11	-		
`pwrite64`	1270	1217 (3%)	741 (47%)	399 (19%)	506 (36		
`pwritev`	10	10 (100%)	10 (10%)	-	10 (20%)		
`sendfile`	1	-	-	1	1		
`sendmmsg`	2	2	2	-	-		
`sendmsg`	12	5 (100%)	3 (100%)	-			
`sendto`	431	389 (7%)	410 (38%)	408 (6%)	-		
`write`	3083	768 (74%)	2877 (91%)	1265 (10%)			
`writev`	754	244 (18%)	742 (76%)	-			

*After initialization, programs typically copy **strings** around **verbatim**.*

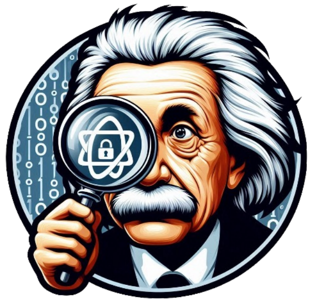
*Many types of syscalls offer many **primitives to an attacker**.*

Evaluation

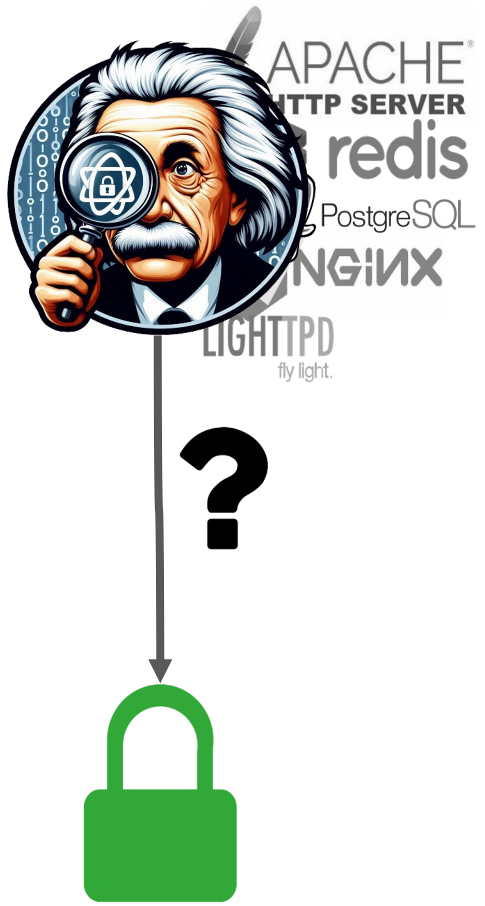
Syscall	Total covered	Total attacker-tainted args. (% that are copied from attacker-controllable data *verbatim*)					
		Arg. 1	Arg. 2	Arg. 3	Arg. 4	Arg. 5	Arg. 6
`execve`	11	7 (86%) *pathname*	7 (86%) *argv*	7 (86%) *envp*			
`mmap`	787	55 (5%)	441 (8%)	55	16 (100%)	17	73
`mprotect`	144	97 (68%)	98 (27%)	1			
`mremap`	11	11	7	11	-		
`pwrite64`	1270	1217 (3%)	741 (47%)	399 (19%)	506 (36		
`pwritev`	10	10 (100%)	10 (10%)	-	10 (20%)		
`sendfile`	1	-	-	1	1		
`sendmmsg`	2	2	2	-	-		
`sendmsg`	12	5 (100%)	3 (100%)	-			
`sendto`	431	389 (7%)	410 (38%)	408 (6%)	-		
`write`	3083	768 (74%)	2877 (91%)	1265 (10%)			
`writev`	754	244 (18%)	742 (76%)	-			

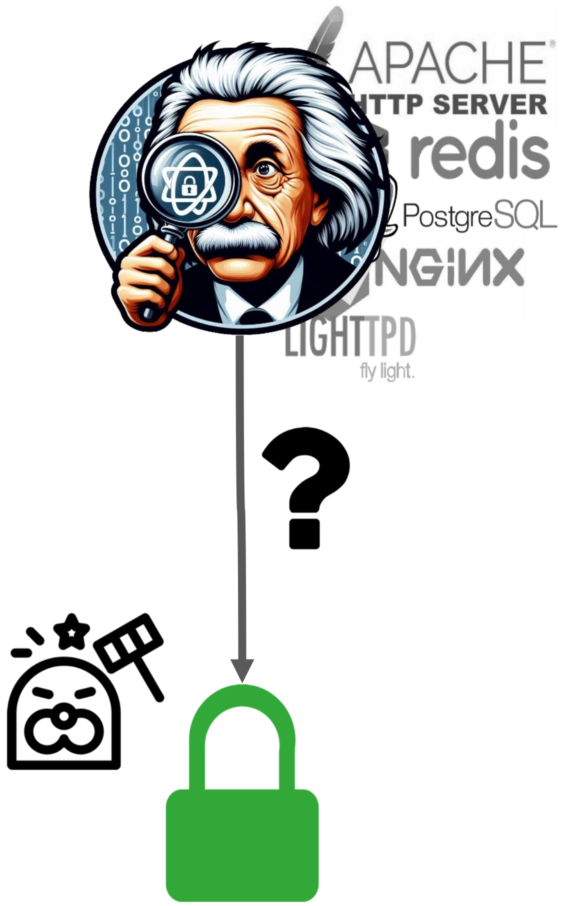
*After initialization, programs typically copy **strings** around **verbatim**.*

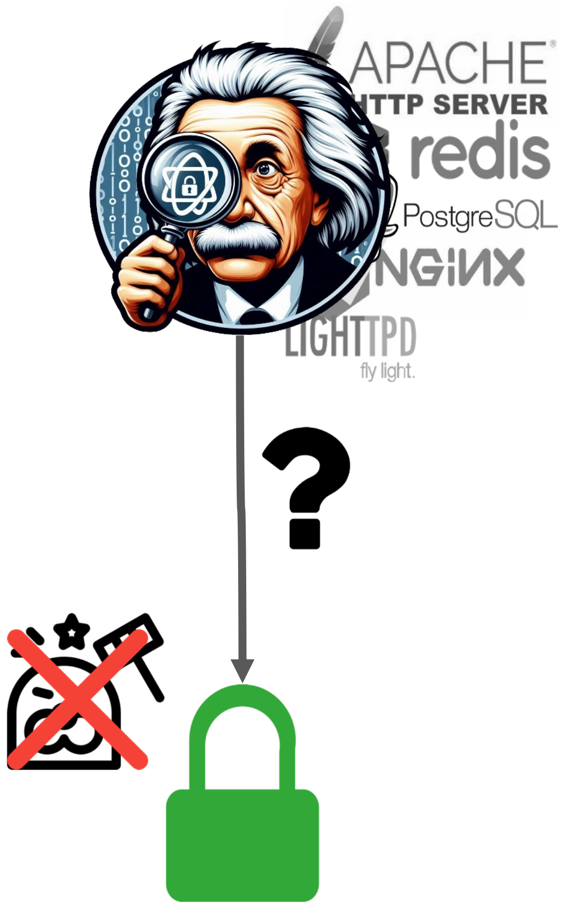
*Many types of syscalls offer many **primitives to an attacker**.*

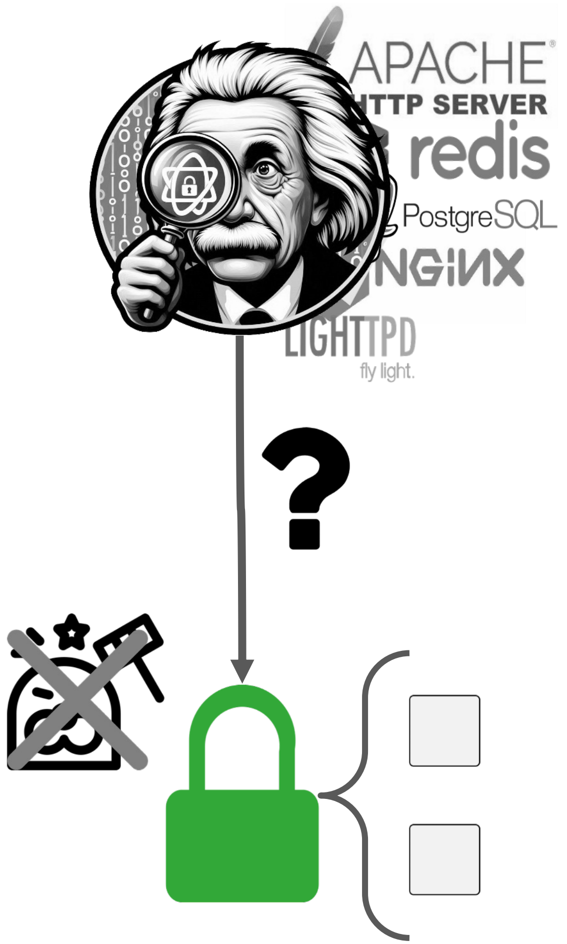


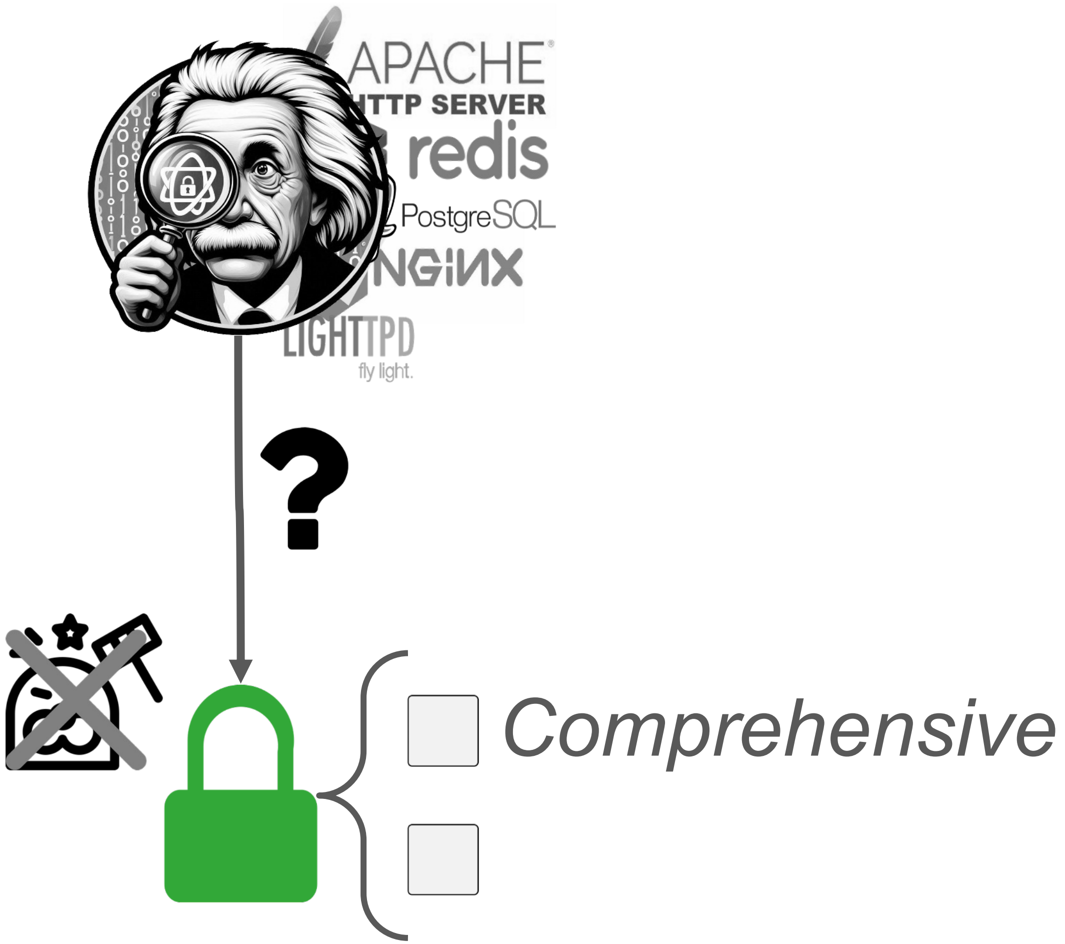
APACHE
HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.


APACHE
HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.
?


APACHE HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.
?


APACHE
HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.
?


APACHE
HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.
?

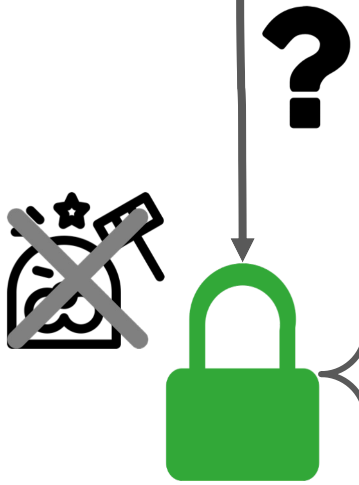

APACHE
HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.
?
☐ Comprehensive
☐

APACHE
HTTP SERVER
redis
PostgreSQL
NGINX
LIGHTTPD
fly light.
?

☐ *Comprehensive*

☐ *Practical*

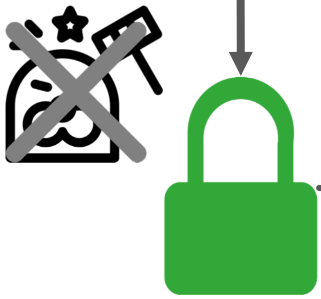
Control-flow hijacking attacks



- ☐ *Comprehensive*
- ☐ *Practical*

Control-flow hijacking attacks

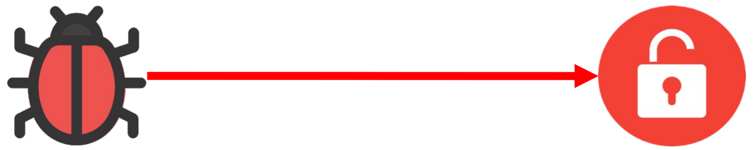
Code pointer



☐ *Comprehensive*

☐ *Practical*

Control-flow hijacking attacks



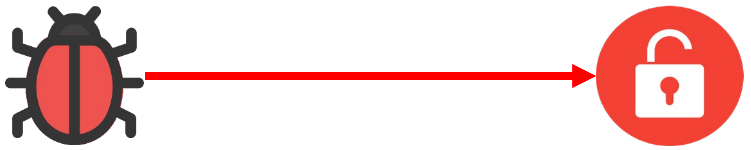
Code pointer

Indirect branch

☐ *Comprehensive*

☐ *Practical*

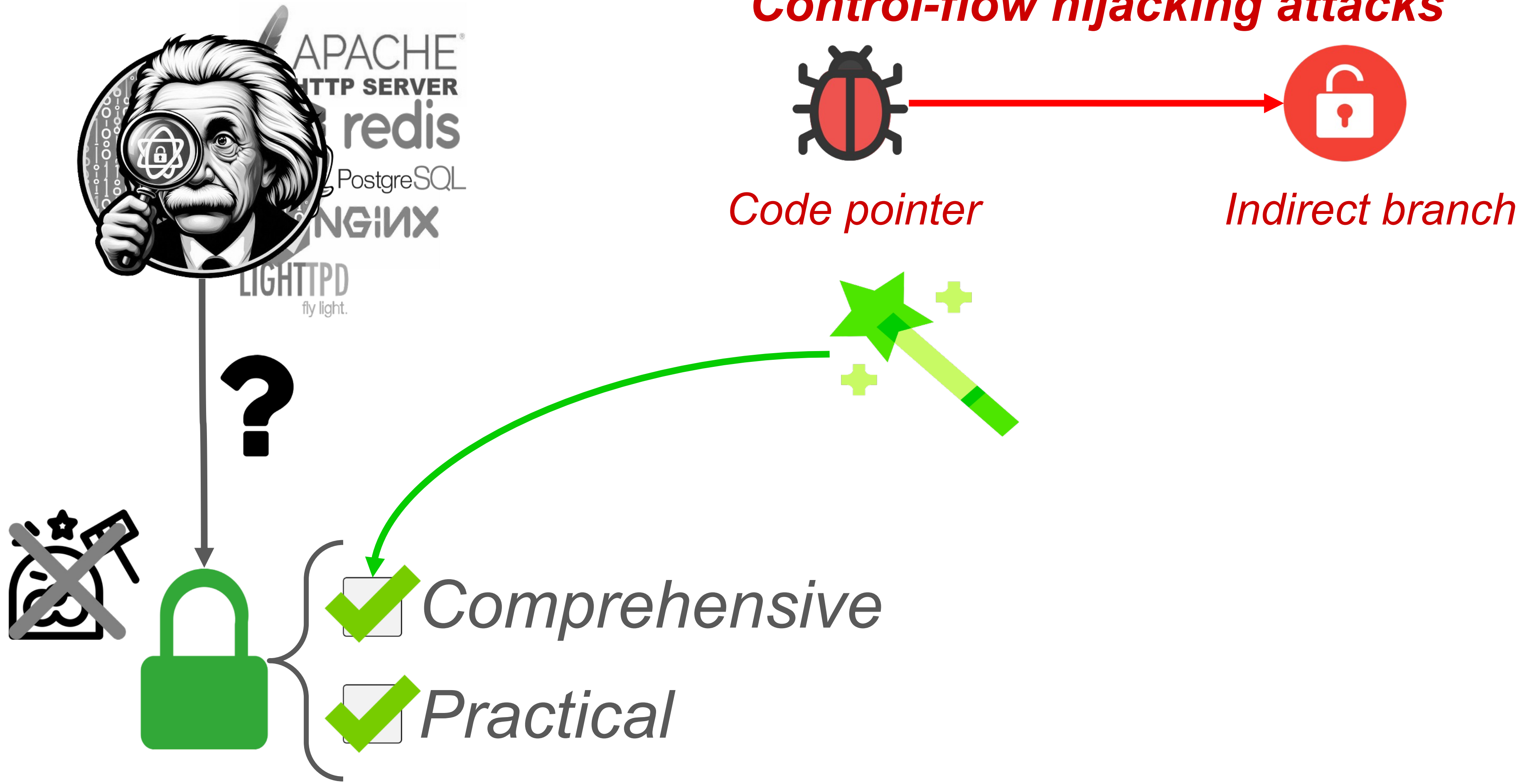
Control-flow hijacking attacks



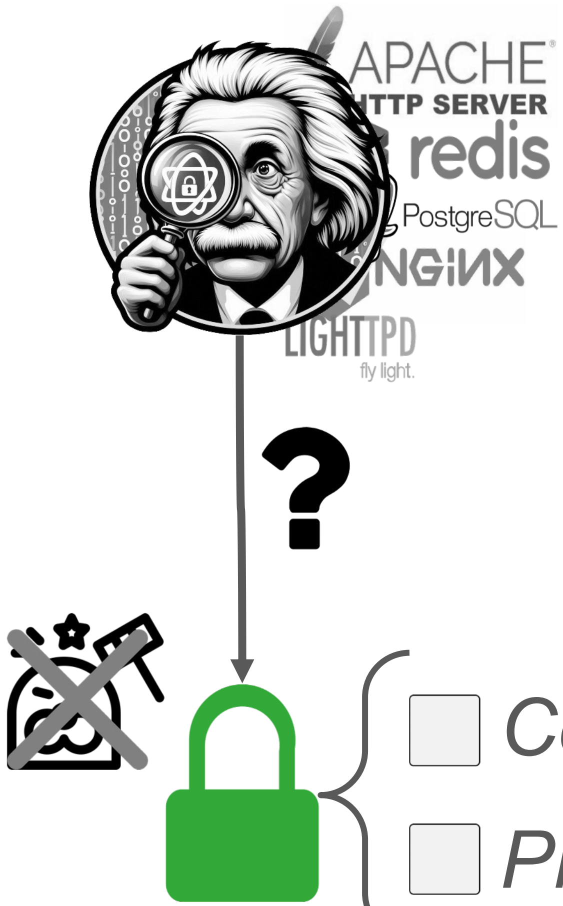
☐ *Comprehensive*

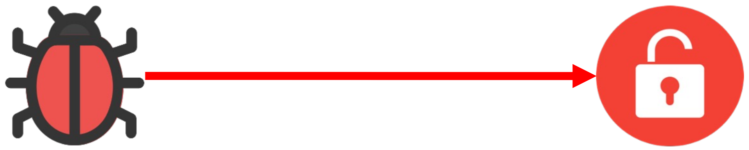
☐ *Practical*

Control-flow hijacking attacks



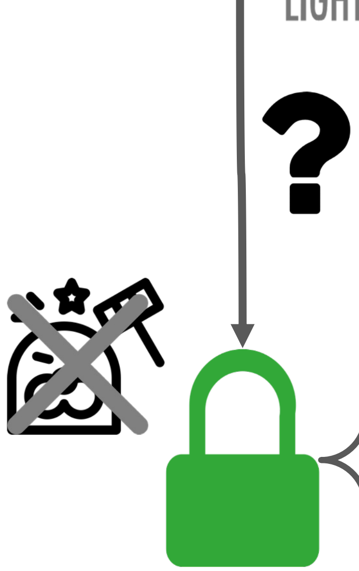
- ✓ Comprehensive
- ✓ Practical





☐ *Comprehensive*

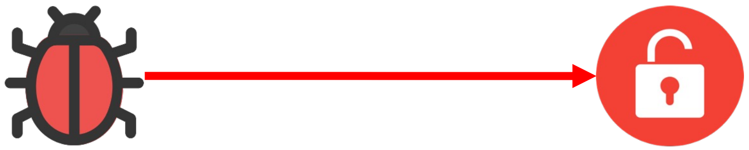
☐ *Practical*



☐ *Comprehensive*

☐ *Practical*

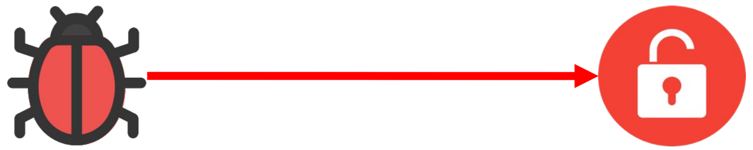
Data-only attacks



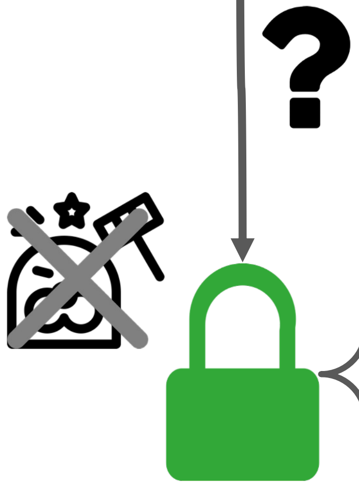
☐ *Comprehensive*

☐ *Practical*

Data-only attacks

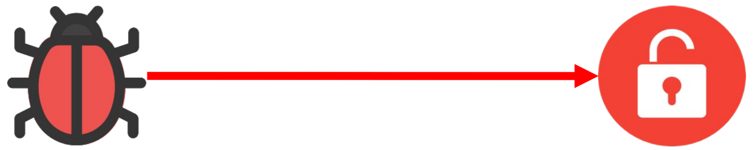


Any string
Any number
Any boolean
…



- ☐ *Comprehensive*
- ☐ *Practical*

Data-only attacks



Any string
Any number
Any boolean
…

Any syscall
Any store
Any cond. branch
…

Data-only attacks

?

Any string	*Any syscall*
Any number	*Any store*
Any boolean	*Any cond. branch*
...	*...*

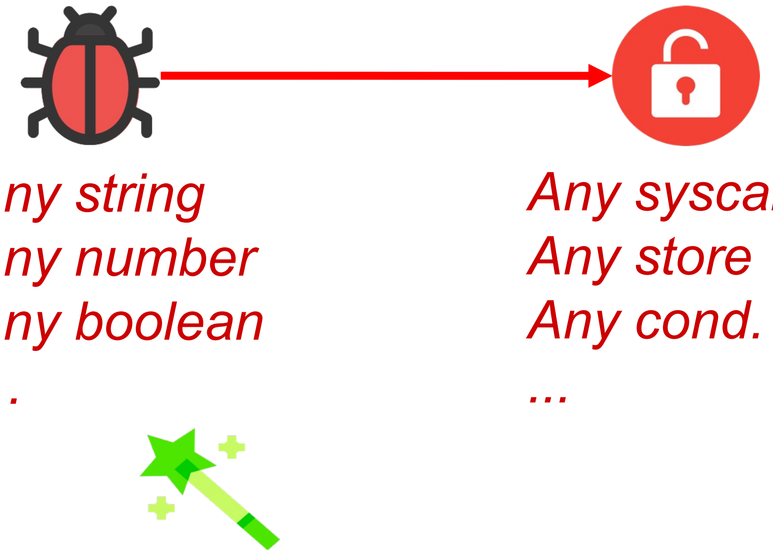
✓ *Comprehensive*

☐ *Practical*

☐ *Comprehensive*

☐ *Practical*

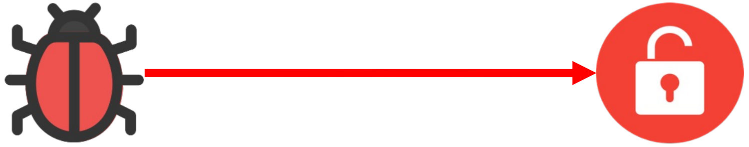
Data-only attacks



Any string
Any number
Any boolean
...

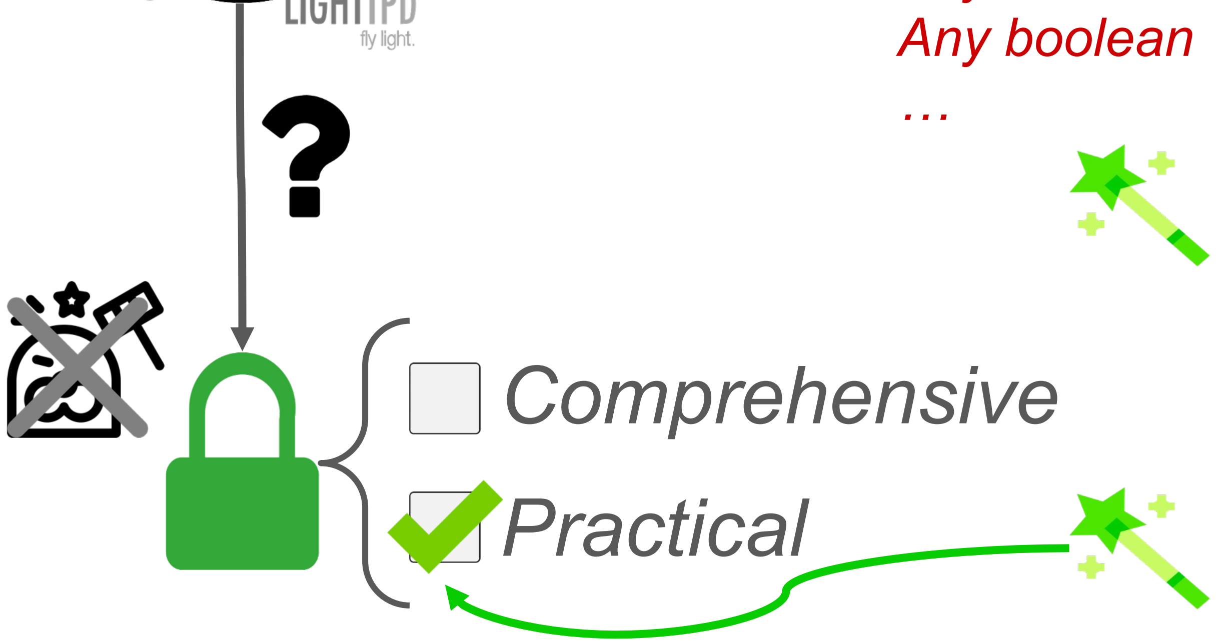
Any syscall
Any store
Any cond. branch
...

Data-only attacks



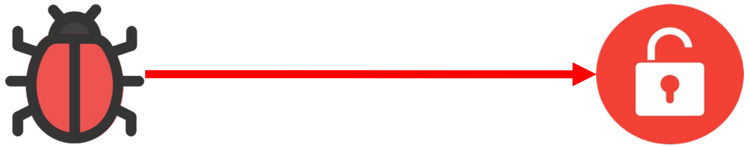
Any string
Any number
Any boolean
...

Any syscall
Any store
Any cond. branch
...



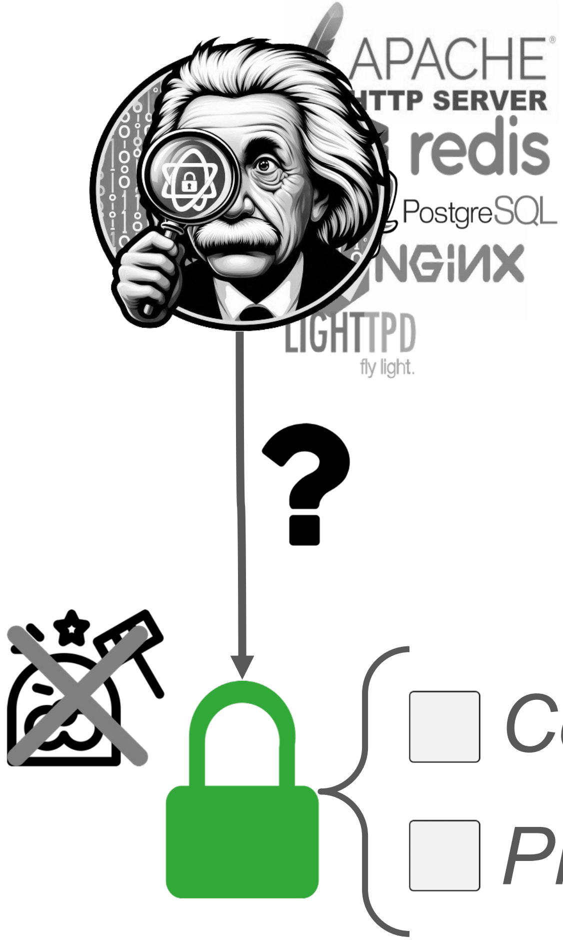
- [ ] Comprehensive
- [✓] *Practical*

Data-only attacks



Any string
Any number
Any boolean
...

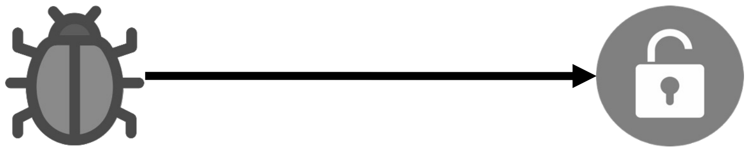
Any syscall
Any store
Any cond. branch
...



☐ *Comprehensive*

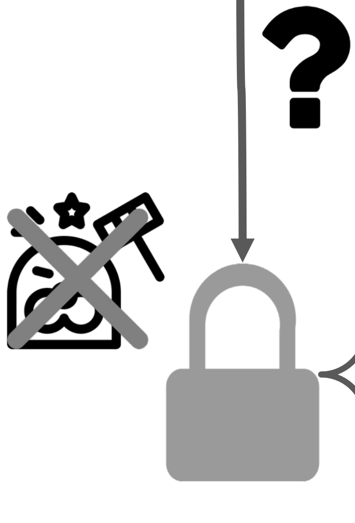
☐ *Practical*

Data-only attacks



Any string
Any number
Any boolean
...

Any syscall
Any store
Any cond. branch
...



☐ *Comprehensive*

☐ *Practical*

We present Einstein, a ***data-only attack*** *generator.*

We present Einstein, a ***data-only attack*** *generator.*

It builds ***hundreds of exploits*** *against popular web servers.*

*We present Einstein, a **data-only attack** generator.*

*It builds **hundreds of exploits** against popular web servers.*

*We call upon researchers and vendors to **rethink mitigation strategies.***