PATENTOVÝ SPIS

(19) ČESKÁ REPUBLIKA

ÚŘAD PRŮMYSLOVÉHO VLASTNICTVÍ

(21) Číslo přihlášky: **1998 - 1141**

(22) Přihlášeno: **20.03.1996**

(30) Právo přednosti:
31.10.1995 US 1995/551260

(40) Zveřejněno: **14.10.1998**
(Věstník č. 10/1998)

(47) Uděleno: **31.07.2001**

(24) Oznámeno udělení ve Věstníku: **12.09.2001**
(Věstník č. 9/2001)

(86) PCT číslo: **PCT/GB96/00664**

(87) PCT číslo zveřejnění: **WO 97/16911**

(11) Číslo dokumentu:

288 937

(13) Druh dokumentu: **B6**

(51) Int. Cl.[7]:
H 04 L 29/06

(73) Majitel patentu:
INTERNATIONALL BUSINESS MACHINES CORPORATION, Armonk, NY, US;

(72) Původce vynálezu:
Gore Robert Cecil, Pflugerville, TX, US;
Haugh John Frederick, Austin, TX, US;

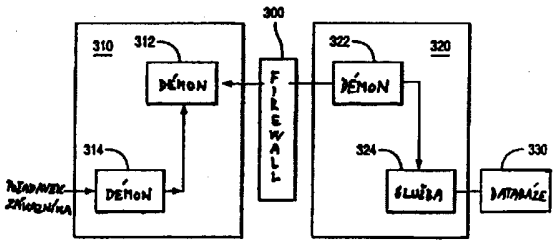
(74) Zástupce:
Kalenský Petr JUDr., Hálkova 2, Praha 2, 12000;

(54) Název vynálezu:

Rozhraní zabezpečené brány

(57) Anotace:

Počítačem implementovaný způsob, jedinečně naprogramovaný počítačový systém a výrobek obsahující počítačem čitelné programové prostředky, umožňuje zákazníkovi na vnějším počítačovém systému (310) zahájit autorizovanou obchodní transakci s použitím vnitřních obchodních prostředků (330) na vnitřní síti (320) bez narušení bezpečnostních firewallů (300). Podrobně způsob řídí vnitřní počítačový systém (320), aby se umožnilo vnějšímu počítačovému systému (310) zahájit požadavek (2) na transakci s použitím vnitřních prostředků (330) bez narušení bezpečnostního firewallu (300) mezi vnitřním počítačovým systémem (320) a vnějším počítačovým systémem (310). Způsob obsahuje první krok ověření spojení zahájeného vnitřním počítačovým systémem (320) mezi vnitřním počítačovým systémem (320) a vnějším počítačovým systémem (310) a tím se založí ověřené spojení. Druhý krok obsahuje volání vnějším počítačovým systémem (310) požadavku (2) na transakci přijatého vnějším počítačovým systémem (310). Odezvou na volání požadavku (2) na transakci obsahuje třetí krok vytvoření vnějším počítačovým systémem (310) řetězce obsahujícího požadavek (2) na transakci, parametry a proměnné prostředí procesu pro spuštění požadavku (2) na transakci. Čtvrtý krok obsahuje vyslání vnějším počítačovým systémem (310) řetězce vnitřnímu počítačovému systému (320) pomocí ověřeného spojení. Pátý krok obsahuje ověření vnitřním počítačovým systémem (320) požadavku (2) na transakci. Šestý krok obsahuje obnovení vnitřním počítačovým systémem (320) původního prostředí procesu. Poslední krok obsahuje spuštění vnitřním počítačovým systémem (320) požadavku (2) na transakci a tím generování výstupu.



CZ 288937 B6

Rozhraní zabezpečené brány

Oblast techniky

Vynález se týká rozhraní zabezpečené brány pro sítě a podrobněji, ale bez omezení, rozhraní zabezpečené brány pro umožnění uživateli z vnější sítě zahájit autorizovanou transakci s využitím vnitřních prostředků bez narušení bezpečnostních firewallů.

Dosavadní stav techniky

Obr. 1 znázorňuje rozhraní zabezpečené brány (SGI) 9 podle předchozího stavu techniky, které má vnější server 4 (např. démon hypertextového transferového protokolu HTTPD) pro zpracovávání požadavků 2 uživatelů/zákazníků od kohokoli ve vnější síti (např. Internetu) a vnitřní server 7 pro zpracovávání požadavků od kohokoli ve vnitřní síti (např. kohokoli, kdo pracuje pro danou společnost) a vnitřní databázi 8. SGI 9 dále obsahuje firewall 6 pro zamezení vnějšího spuštění vnitřních transakcí na vnitřních databázích 8. Firewall 6 tudíž zabraňuje vnějším zákazníkům v zahájení přímého spojení s vnitřní sítí (tj. s vnitřním serverem 7 a vnitřními databázemi 8). Toto omezení zabraňuje platným transakcím jako například požadavkům na zakoupení produktů nebo služeb, protože zákazníci prostě nemohou spustit vnitřní transakci z vnější sítě. Takový konvenční firewall je popsán v IEEE Communications magazine vol. 33 č. 9, září 1994, US, strany 50-57, S. M. Bellowin a kolektiv.

Konvenční řešení výše popsaného problému vyžaduje otevření konkrétního portu (např. portu 84) ve firewallu 6 pro příchozí data. Toto řešení však jasně ponechává vnitřní síť vystavenou vnějšímu útoku. Jiné řešení umísťuje všechny požadované prostředky (např. databáze 8) na vnější server 4. Toto řešení však i nadále zamezuje spouštění vnitřních transakcí. Vnější server 4 dále nemusí mít dostatek úložného prostoru k umístění všech požadovaných prostředků, nebo mohou být prostředky příliš důvěrné, než aby mohly být umístěny na vnějším serveru (např. uživatelská data), což omezuje služby, které mohou být poskytnuty.

Existuje tudíž velká poptávka techniky, která umožňuje zákazníkovi zahájit ověřenou obchodní transakci s využitím vnitřních obchodního prostředků bez narušení zabezpečovacích firewallů.

Podstata vynálezu

Počítačem implementovaný způsob, jedinečně naprogramovaný počítačový systém a výrobek obsahující počítačem čitelné prostředky tudíž umožňují zákazníkovi ve vnější síti spouštět ověřenou obchodní transakci s použitím vnitřních obchodních prostředků na vnitřní síti bez narušení bezpečnostních firewallů.

Konkrétně způsob řídí vnitřní počítačový systém tak, aby umožnil vnějšímu počítačovému systému spustit požadavek na transakci s použitím vnitřních prostředků bez narušení bezpečnostního firewallu mezi vnitřním počítačovým systémem a vnějším počítačovým systémem. Způsob obsahuje první krok ověření spojení zahájeného vnitřním počítačovým systémem mezi vnitřním počítačovým systémem a vnějším počítačovým systémem, čímž se založí ověřené spojení. Druhý krok obsahuje volání vnějším počítačovým systémem, požadavek na transakci přijatý vnějším počítačovým systémem. Odezvou na volání požadavku na transakci obsahuje třetí krok vytvoření vnějším počítačovým systémem řetězce, který obsahuje požadavek na transakci a proměnné prostředí procesu pro spuštění požadavku na transakci. Čtvrtý krok obsahuje vyslání vnějším počítačovým systémem řetězce do vnitřního počítačového systému ověřeným spojením. Pátý krok obsahuje ověření vnitřním počítačovým systémem požadavku na transakci. Šestý krok

obsahuje obnovení vnitřním počítačovým systémem původního prostředí procesů. Poslední krok obsahuje spuštění vnitřním počítačovým systémem požadavku na transakci a tím generování výstupu.

Předmětem vynálezu je tudíž vytvořit rozhraní zabezpečené brány, které je transparentní pro uživatele a existující transakční program.

Dalším předmětem tohoto vynálezu je umožnit uživateli platně spustit transakci přes firewall ve vnitřní síti s použitím vnitřních prostředků.

Ještě dalším předmětem vynálezu je umožnit uživateli spustit pouze platnou skupinu autorizovaných transakcí.

Ještě dalším předmětem je bezpečně autorizovat spojení mezi vnitřním počítačovým systémem a vnějším počítačovým systémem předtím, než vnější počítačový systém přijme požadavky na transakce od uživatelů.

Ještě dalším předmětem je uložit transakční programy dovnitř firewallu bez nutnosti jejich úpravy.

Přehled obrázků na výkresech

Vynález bude blíže vysvětlen prostřednictvím konkrétních příkladů provedení znázorněných na výkresech, na kterých představuje

obr. 1 blokové schéma konvenčního síťového systému pro použití v implementaci tohoto vynálezu;

obr. 2 vzorová hardwarová konfigurace pro implementaci tohoto vynálezu;

obr. 3 blokové schéma rozhraní zabezpečené brány (SGI) podle upřednostňovaného provedení; a

obr. 4 podrobnější blokové schéma procesu SGI výše znázorněného na obr. 3.

Příklady provedení vynálezu

Upřednostňované provedení obsahuje počítačem implementovaný způsob, jedinečně naprogramovaný počítačový systém a podrobnou logiku včleněnou do paměti pro směrování vnitřního počítačového systému k umožnění vnějšímu uživateli/zákazníkovi spustit autorizovanou obchodní transakci s využitím vnitřních obchodních prostředků bez narušení bezpečnostních firewallů.

Tento vynález je vyzkoušen na počítačovém systému znázorněném na obrázku 2. Počítačový systém 100 obsahuje hlavní procesorovou jednotku (CPU) 10, jako například IBM PowerPC 601 nebo mikroprocesor Intel 486 pro zpracování vyrovnávací paměti 15, paměti 14 s přímým přístupem (RAM), permanentní paměti 16 a paměti RAM nezávislé na energetickém zdroji (NVRAM) 32. jeden nebo více disků 20, řízených I/O adaptérem 18 zajišťují dlouhodobý úložný prostor. Lze použít množství jiných úložných médií včetně pásek, jednotek CD-ROM a WORM. Odstranitelná média mohou být také zajištěna k ukládání dat nebo počítačových instrukcí pro zpracování. (IBM a PowerPC jsou obchodní značky International Business Machines Corporation a Intel je obchodní značka Intel Corporation).

Instrukce a data z počítače s jakýmkoli vhodným operačním systémem, jako například Sun Solaris, Microsoft Windows NT, IBM OS/2 nebo Apple System 7, řídí CPU 10 z RAM 14. Počítač tudíž spouští z RAM 14. v upřednostněném provedení však na IBM RISC Systému/6000 běží operační systém AIX, což je implementace společnosti IBM Corporation operačního systému UNIX. Jak bylo však výše popsáno, lidé znalí v oboru ihned poznají, že k implementaci tohoto vynálezu mohou být použity jiné platformy a operační systémy. (Solaris je obchodní značka Sun Microsystems Inc., Windows Nt je obchodní značka Microsoft Corporation, System 7 je obchodní značka Apple Computer Corporation a OS/2, RISC System/6000 a AIX jsou obchodní značky IBM Corporation. UNIX je registrovaná obchodní známka ve Spojených státech a v jiných zemích povolená výlučně společností X/Open Company Limited).

Uživatelé komunikují s počítačovým systémem 100 I/O zařízením (tj. uživatelskými ovládacími prvky) řízeným adaptérem 22 uživatelského rozhraní. Zobrazovací jednotka 38 zobrazuje informace uživateli, zatímco klávesnice 24, ukazovací zařízení 26 a reproduktor 28 umožňují uživateli řídit počítačový systém. Komunikační adaptér 34 řídí komunikaci mezi tímto počítačovým systémem a jinými zpracovávajícími jednotkami připojenými do sítě. Zobrazovací adaptér 36 řídí komunikaci mezi tímto počítačovým systémem a zobrazovací jednotkou 38.

Obr. 3 znázorňuje blokové schéma a průběh procesu rozhraní zabezpečené brány (SGI) podle upřednostňovaného provedení. SGI se nachází na dvojici vnějších počítačových systémů 310 a 320 (serverů), přičemž každý je implementován na počítačovém systému 100 (viz. obr. 2). Vnější počítačový systém 310 se nachází vně firewallu 300, zatímco vnitřní server 320 se nachází uvnitř firewallu 300. Firewall 300 je implementován s použitím jakéhokoli vhodného konvenčního firewallu, který zabraňuje v průchodu vnějších transakcí skrz něj do vnitřního serveru 320. v upřednostňovaném provedení je firewall 300 síťový směrovač (např. Cisco směrovač). Odborník však ihned pozná, že firewall 300 by se mohl nacházet na vnitřním serveru 320.

Vnější počítačový systém 310 spravuje komunikaci s uživateli/zákazníky ve vnější síti, jako například na Internetu. Odborník si však uvědomuje, že lze použít jakéhokoli typu komunikačního protokolu, jako například SNA nebo X.25 na jakékoli veřejné nebo soukromá síti. Vnitřní server 320 spravuje komunikaci vnitřních prostředků (např. databáze 330) ve vnitřní síti, jako například ve vnitřní podnikové informační síti. Na vnějším počítačovém systému 310 běží vnější démon 312, zatímco na vnitřním serveru 320 běží třetí démon 322, což umožňuje komunikaci firewallem 300. Démon je dlouhodobě běžící počítačový program, který čeká na vnější události a spouští předem definované skupiny akcí, kdykoli tyto události nastanou. Démoni poslouchají požadavky na služby a pokud tyto požadavky existují, provedou je. Na vnějším počítačovém systému 310 také běží démon 314, který poslouchá požadavky na služby z vnější sítě. Vnitřní server 320 obsahuje obslužný program 324 pro spouštění požadované vnitřní transakce. Obslužný program 324 a vnitřní databáze 330 představují skupinu počítačových programů, které implementují obchodní transakci (zde podrobněji popsanou).

Obr. 4 znázorňuje podrobnější schéma průběhu procesu SGI výše znázorněného na obr. 3. Vnější počítačový systém 310 obsahuje jakýkoli vhodný konvenční démon 314 komunikačního protokolu, cgi-bin 415, rutinu 416 sgi_client a vnější démon 312. Vnější démon 312 obsahuje software klient-server pro komunikaci s rutin 416 sgi_client a třetí démon 322. Rutina 416 sgi_client obsahuje software klient-server pro komunikaci s vnějším démonem 312. Cgi-bin 415 je adresář softwaru, který je spouštěn démonem 314. Konkrétně v tomto případě obsahuje cgi-bin 415 example.pl, což je speciální perl skript pro komunikaci s rutinou 416 sgi_client (zde podrobněji popsanou). V upřednostňovaném provedení je démon 314 konvenční démon Hypertextového Transferového Protokolu (httpd) (také běžně známého jako web server).

Vnitřní server 320 obsahuje třetí démon 322, obslužný program 324 a cgi-bin 426. Obslužný program 324 komunikuje s vnějším démonem 312, třetím démonem 322 a spouští rutiny cgi-bin (např. example.pl 480). V tomto příkladě komunikuje example.pl 480 s vnitřními podnikovými

databázemi (např. podnikovou databází 330 na obr. 3), aby se ověřil uživatel/zákazník a spustila obchodní transakce.

Předtím, než může zákazník/uživatel úspěšně žádat o transakci na 410, vnitřní server 320 a vnější počítačový systém 310 se musí správně spojit. Aby se tak mohlo stát, spustí vnější operační systém vnější démon 312, přičemž mu identifikuje komunikační port a umístění souboru s hesly, který se nachází v systému souborů (není ukázán) na vnějším počítačovém systému 310. Poté vnější démon 312 přečte osmiznakové heslo ze souboru s hesly, vytvoří soket na identifikovaném komunikačním portu a poslouchá na tomto soketu volání spojení od třetího démona 322. Vnější démon 312 tudíž přijímá roli serveru a čeká na 430 na volání o připojení od třetího démona 322, který přijímá roli klienta. Dále vnější démon 312 vytvoří soket na druhém portu (démon 312 komunikační port + 1) a čeká na pokus o spojení od rutin 416 sgi_client na 432 (zde popsáno podrobněji).

Vnitřní operační systém spustí třetího démona 322, přičemž mu identifikuje komunikační port pro připojení třetího démona 322 s vnějším démonem 312, název uzlu vnějšího počítačového systému 310, umístění souboru s hesly nacházejícího se v systému souborů (není ukázán) na vnitřním serveru 320 a umístění platného obslužného souboru nacházejícího se v systému souborů (není ukázán) na vnitřním serveru 320. Třetí démon 322 dále přečte osmiznakové heslo ze souboru s hesly, přečte obslužný soubor a uloží tabulku platných služeb do paměti, vytvoří soket na identifikovaném komunikačním portu a nakonec generuje standardní volání o připojení na 450 přes firewall 300 vnějšímu démonu 312, který poslouchá na 430. Protože spojení bylo zahájeno od vnitřního serveru, firewall 300 povolí spojení.

Poté, co se třetí démon 322 a vnější démon 312 spojí, třetí démon 322 a vnější démon 312 se musí vzájemně důkladně ověřit. Aby se tak mohlo stát, třetí démon 322 zahájí volání vnějšího operačního systému za účelem získání aktuální časové značky, vyšle tuto časovou značku vnějšímu démonu 312 a čeká jako odezvu na ověřovací řetězec. Vnější démon 312 přijme časovou známku a vytvoří ověřovací řetězec úpravou (změnou, popsána níže) svého osmiznakového hesla časovou známkou poskytnutou třetím démonem 322, přičemž zakóduje tento pozměněný znakový řetězec standardním UNIXOvým příkazem crypt (nebo jakýmkoli vhodným šifrovacím algoritmem, jako například DES) a pak vyšle výsledný ověřovací řetězec třetímu démonu 322 na 431. Následující program v jazyce C znázorňuje proces úpravy osmiznakového hesla časovou značkou. Tento kód "create_auth" vyžaduje tři parametry - první je časová značka (tj. auth_time), druhý je heslo (tj. "cred", což je ukazatel na heslo) a třetí je vyrovnávací paměť k uložení vygenerovaného ověřovacího řetězce:

```
int create_auth (time_t, char * cred, char * p)
{
        char buf [9];       /*dočasná vyrovnávací paměť*/
        int i;

        bzero (buf, sizeof(buf)); /*vyčistit vyrovnávací paměť*/
        strcpy (buf, cred);                 /*zavést do vyrov. paměti heslo*/

/*úprava jednotlivých znaků ve vyrovnávací paměti*/

for (i = 0; i < 8; i++)          {
        buf [i] ^= (auth_time & 0177); /*logický AND časové značky,
        pak výlučný OR výsledku z jednotlivými znaky ve vyrovnávací
        paměti; s každou iterací úprava časové značky*/
        auth_time >>=4;       /*bitový posun časové značky*/

{
```

```
for (i = 0; i < 8; i++)
      if (buf [i] = = 0)        /*protože platný řetězec znaků nemůže obsahovat NULL, změnit
                                všechny NULL na 1 */
          buf [i] = 1;

strcpy (p, crypt (buf, aa") + 2)  /*zašifrovat vyr. paměť s použitím klíče aa*/
                                /*přeskočit první dva znaky výsledku šifrování (což je
                                klíč aa)*/
                                /*zkopírovat výsledek šifrování do uživatelem dodané
                                vyr. paměti, na kterou ukazuje P*/
return 0;
}
```

Třetí démon 322 podobně upraví své heslo časovou značkou, zašifruje jej a porovná jej s ověřovacím řetězcem dodaným vnějším démonem 312. Pokud se ověřovací řetězec shoduje, je proces obrácen a vnější démon 312 podobně ověří třetí démon 322 (tj. získá novou časovou značku od vnějšího operačního systému, vyšle tuto časovou značku třetímu démonu 322, třetí démon 322 upraví své heslo novou časovou značkou, zašifruje jej a vyšle jej zpět vnějšímu démonu 312 k ověření).

Tento ověřovací proces používá osmiznakové heslo, které je známé jak vnějším, tak vnitřním démonům 312 a 322, funkci úpravy znaků znáhodněnou časovou značkou a šifrovací proces. Kvůli funkci úpravy produkuje výše uvedený proces různé zašifrované ověřovací řetězce pro každé ověření a každou transakci. To významně snižuje jeho zranitelnost vůči útoku, protože zachycený ověřovací řetězec je bezcenný pro jakékoli následující transakce.

Poté co se třetí démon 322 a vnější démon 312 vzájemně ověřili, třetí démon 322, který předtím pracoval jako klient, nyní přebírá roli serveru a čeká na 452 na vnější démon 312 až poskytne obslužný řetězec na 453. Vnější démon 312 vytvoří jiný soket na druhém zadaném portu a čeká (poslouchá) na pokus o spojení od rutiny 416 sgi_client na 432. Proto vnější démon 312 přebírá duální roli pseudo klienta vzhledem ke třetímu démonu 322 (informace jsou posílány mezi nimi) a serveru vzhledem k rutině 416 sgi_client.

Démon 314 je nyní připraven přijmout požadavek 410 zákazníka. Požadavek zákazníka by mohla být například transakce koupě průzkumu informací o konkrétní akcii nebo o trhu s penězi. Na 410 se zákazník rozhodne spustit následující požadavek na transakci:

http: //external_server/cgi_bin/example.pl?akcie1+akcie2
klepnutím na konkrétní ikonu nebo zvýrazněnou větu v systému zákazníka, ve kterém běží uživatelské rozhraní aplikace http klienta. Uživatelské rozhraní http klienta se typicky dotáže uživatele na podrobnější informace o transakci (např. o kterou akcii nebo peněžní trh se jedná) stejně tak jako na informace týkající se placení (např. na číslo kreditní karty). Uživatel může být rovněž vyzván k vložení svého id uživatele a hesla pokud je požadovaná služba poskytována pouze autorizovaným uživatelům.

Formát vyslaného vstupu od uživatele závisí na typu formuláře jazyka Hyper Text Markup Language (HTML) použitého k implementaci transakce. Existují dva typy konvenčních HTML formulářů:

Formulář "GET" umístí veškerý vstup uživatele na příkazový řádek. Proto se akcie1, akcie2 a jakýkoli jiný vstup uživatele stane součástí příkazového řádku:

.../cgi-bin/example.pl?akcie1+akcie2+číslokreditkarty+datumvypršení

Protože však bude příkazový řádek vyslán sítí jako jednoduchý text, není vhodné přenášet číslo kreditní karty zákazníka a datum vypršení po síti. Proto je použit formulář "PUT" HTML se šifrováním tak, aby bylo číslo kreditní karty a datum vypršení bezpečně poslány po síti. Po poskytnutí všech těchto informací pošle aplikace http klienta požadavek prostřednictvím http vnějšímu počítačovému systému 310 na 410.

Na 460 ověřuje démon 314 heslo zákazníka podle běžně známé a instalované HTPP ověřovací techniky (např. šifrování hesla zákazníka standardním příkazem UNIXu crypt a porovnáním výsledku s položkou hesla v souboru s http heslem umístěném v démonu 314). Pokud je id uživatele a heslo platné, na 461, démon 314 rozezná formulář "PUT", automaticky rozšifruje proud znaků a vytvoří příslušné prostředí UNIXOvého procesu. Démon 314 obsahuje konvenční, běžně známý http konfigurační soubor (není ukázán) pro vytváření standardního prostředí UNIXOvých procesů včetně proměnných PATH, USERNAME, LOGNAME a AUTHTYPE. Později httpsvc 470 znovu vytvoří toto prostředí procesu na 471 (zde popsáno). Jakmile bylo prostředí procesu vytvořeno, spustí démon 314 example.pl (který my se měl nacházet v cgi-bin 415), přičemž mu vyšle jakékoli požadované parametry (např. akcie1 a akcie2) a uživatelský vstup do standardního vstupního proudu example.pl.

Za předpokladu, že se example.pl nenachází v cgi_bin 415, pokud by firewall 300 neexistoval, example.pl by přímo komunikoval s vnitřní databází 330 (viz. obr. 3) a vykonal požadovanou transakci. Protože však firewall 300 existuje a zabraňuje example.pl v přímé komunikaci s vnitřní databází 330, example.pl není aktuální transakční program. Současný transakční program se spíše nachází v cgi-bin 426 jako example.pl 480, který je uvnitř firewallu 300. Cgi-bin 415 tudíž obsahuje "speciální" perl skripty (např. example.pl), které jsou spouštěny s použitím stejného příkazu, který by spustil aktuální transakční programy, které se nacházejí v cgi-bin 426. Eventuelně pokud vnější počítačový systém 310 nabízí více služeb, přičemž každá z nich bude požadovat "speciální" perl skript k zavolání rutiny 416 sgi_client stejným způsobem, example.pl by mohl být symbolický odkaz (tj. nepřímý odkaz na název souboru) na jeden perl skript nacházející se v cgi-bin 415. Důležité je, že požadavky dostupné zákazníkovi jsou omezeny na perl skripty a odpovídající transakční programy nacházející se v cgi_bin 415 a cgi_bin 426 v uvedeném pořadí.

Perl skript example.pl umísťuje všechny parametry jemu poslané od démona 314 do prostředí procesu (např. SGIARG1=akcie1; SGIARG2=akcie2), umísťuje svůj název (název kterým byl volán, v tomto případě example.pl) do prostředí procesu (např. SGICMD=example.pl), spouští UNIXOvý příkaz env (který vypisuje proměnné prostředí procesu) a nakonec umísťuje všechny proměnné prostředí procesu do řetězce záhlaví. Řetězec záhlaví se nyní objeví jako například:

"PATH-/bin:/usr/bin\nAUTHTYPE=PEM\nUSERNAME=JohnDoe\nSGIARG1
=akcie1\nSGIARG2=akcie2\=nSGICMD=example.pl").

Dále, na 463, perl skript example.pl zavolá vnější operační systém, aby získal označený druhý port (komunikační port + 1 démona 312), spustí rutinu 416 sgi_client, přičemž jí vyšle typ požadované služby (např. httpsvc), určený druhý port, název uzlu vnějšího serveru, řetězec záhlaví a id uživatele zákazníka. Example.pl také vyšle jako standardní vstup jakýkoli standardní vstupní znakový proud (např. text uživatelského vstupu) rutině 416 sgi_client. Později pošle example.pl jakýkoli výstup přijatý od rutiny 416 sgi_client démonu 314 na 469.

Jakmile se rutina 416 sgi_client spustí s použitím informací do ní vyslaných na 463, založí rutina 416 sgi_client ověřené spojení s vnějším démonem 312. Aby se tak stalo, na 417 přečte rutina 416 osmiznakové heslo ze soukromého klientského souboru s hesly (není ukázán), který se nachází na vnějším počítačovém systému 310, a založí spojení s vnějším démonem 312 na určeném druhém portu, který poslouchá na 432 z druhého soketového spojení. Na 433 vytvoří vnější démon 312 svou kopii a spustí ji. (např. UNIXOvý proces fork). Rodičovský proces dá

soketové spojení odvozenému procesu a vrátí se na 430, aby poslouchal další volání ze třetího démona 322.

Na 434 ověřuje odvozený proces rutinu 416 sgi_client. Aby se tak mohlo stát, vnější démon 312 také přečte osmiznakové heslo ze soukromého klientského souboru s hesly (není ukázán), který se nachází na vnějším počítačovém systému 310. Vnější démon 312 zahajuje volání vnějšího operačního systému, aby obdržel aktuální časovou značku, vyšle časovou značku rutině 416 sgi_client na 432 a čeká jako odpověď na ověřovací řetězec. Rutina 416 sgi_client přijme časovou značku a vytvoří ověřovací řetězec úpravou svého osmiznakového hesla pomocí časové značky dodané vnějším démonem 312, přičemž zašifruje tento upravený řetězec znaků standardním UNIXOvým příkazem crypt a pak vyšle výsledný ověřovací řetězec vnějšímu démonu 312 na 434. Vnější démon 312 podobně upraví své heslo pomocí časové značky, zašifruje jej a porovná jej s ověřovacím řetězcem poskytnutým rutinou 416 sgi_client. Pokud se ověřovací řetězec shoduje, rutina 416 sgi_client je ověřena.

Na 419 pokud je ověření úspěšné, rutina 416 sgi_client vyšle typ požadované služby vnějšímu démonu 312. V tomto příkladě rutina 416 sgi_client vždy vyžaduje HTTP službu, protože rutina 416 sgi_client byla nepřímo volána HTTP démonem 314. Speciální perl skript (tj. Example.pl) předtím spustil rutinu 416 sgi_client s použitím parametru udávajícího, že požadovaná služba je "httpsvc". Vnější démon 312 pak vyšle "httpsvc" požadavek na službu třetímu démonu 322 na 435.

Na 452 čeká třetí démon 322 na požadavek na službu, který se má přijmout z vnějšího démona 312. Na 453 přijme třetí démon 322 požadavek na službu od vnějšího démona 312, vytvoří svůj duplikovaný obraz a spustí jej (např. UNIXOvý proces fork). Rodičovský proces dává síťové soketové spojení odvozenému procesu a vrátí se na 450, aby zahájil jiné spojení s vnějším démonem 312. Na 454 ověří odvozený proces požadovanou službu podle seznamu platných spustitelných služeb (např. httpsvc), který se nachází v tabulce v paměti a úplnou cestou adresářů k těmto službám. Pokud požadovaná služba není v seznamu platných služeb, bude odepřena. Tudíž dokonce i pokud by neautorizovaný uživatel získal přístup vnějším démonem 312 k vnitřnímu démonu, bude omezen na služby nacházející s v seznamu platných služeb.

Pokud je požadavek na službu platný, na 455, třetí démon 322 zavolá UNIXOvý příkaz exec (tj. překryje se novým obslužným programem spustí) požadovaný příkaz a dá síťové soketové spojení httpsvc 470. Httpsvc 470 přidá jednu dodatečnou proměnnou prostředí do prostředí procesu, což je název vnějšího démona 312. SGI přidá další proměnnou prostředí, takže example.pl 480, může, je-li potřeba, určit, že SGI spustilo example.pl 480 spíše než http démon 314.

Jako poznámka na okraj, vnější démon 312, třetí démon 322, rutina 416 sgi_client a httpsvc 470 mají každá záznamové účtovací a chybové soubory. Každá má ladicí a trasovací parametry, které způsobují umísťování různého množství informací do záznamů o chybách a účtování. Navíc pokud je trasovací parametr nastaven rutinou 416 sgi_client, pak vnější démon 312, třetí démon 322 a httpsvc 470 budou všichni trasovat tuto konkrétní transakci ve svých příslušných záznamových souborech s chybami bez ohledu na to, jak bylo nastaveno trasování při původním spuštění každého z nich.

Na 436 vysílá vnější démon 312 předem vytvořené záhlaví obslužnému programu 324, který jej přijme na 471. Odezvou obslužný program 324 rozloží záhlaví (které obsahuje původní proměnné prostředí procesů) do řetězců proměnná=hodnota a obnoví původní prostředí procesu definované v example.pl. Obslužný program 324 zjistí příslušný program, který se má zavolat v cgi-bin 426 z proměnné záhlaví SGICMD=example.pl, vytvoří komunikační kanály (například kolony) ke komunikaci s example.pl 480 a zavolá example.pl 480 na 472. Na 437 vyšle vnější

démon 312 standardní vstupní znakový proud (např. text) obslužnému programu 324. Na 473 vyšle obslužný program 324 text do standardního vstupu example.pl 480.

V tomto bodě, protože obslužný program 324 obnovil původní prostředí procesu na 471 (které bylo původně vytvořeno v kroku 462), věří example.pl 480,že je spouštěn na 472 http démonem 314, spíše nežli pomocí SGI (ačkoli volitelně může zjistit, že SGI jej zavolalo z další proměnné prostředí, kterou k záhlaví přidal httpsvc 470). Podle toho je SGI transparentní jak k zákazníkovi, k http démonu 314, tak k aktuálnímu transakčnímu programu, který se nachází v example.pl 480. Tudíž není potřeba měnit ani http démon 314, ani transakční program nacházející se v example.pl 480.

Veškeré informace jsou nyní přítomny pro example.pl 480 ke spuštění vnitřní transakce na databázi 330 na 481. Jakmile je transakce dokončena (ať už úspěšně, či neúspěšně), je na 481 výstup z transakce vrácen zákazníkovi. Na 482 obdrží example.pl 480 výstup z transakce a vyšle jej do kolony 474 obslužného programu 324. Na 474 obslužný program 324 vyšle výstup do vnějšího démona 312. Na 438 vyšle vnější démon 312 výstup rutině 416 sgi_client. Na 464 rutina 416 sgi_client vyšle výstup do speciálního perl skriptu example.pl. Na 465 vyšle example.pl výstup démonu 314. Na 466 vyšle démon 314 výstup zákazníkovi.

Zákazníkem zahájená transakce tudíž muže být bezpečně vyslána z démona 314 do vnějšího démona 312, z vnějšího démona 312 do třetího démona 322 pro ověření na 454 a zpracování na 481 a nakonec může být výstup vrácen zákazníkovi na 466. Požadavek a text uživatele jsou zpřístupněny pomocí firewallu 300 vnitřnímu transakčnímu zpracování, přičemž všechny jsou plně pod kontrolou SGI, stále však pro zákazníka plně transparentní. Protože třetí démon 322 provádí ověření na 451, vyloženě vnutí služby dostupné vnější síti na 454 a volitelně provede ověření uživatele na 481, vystavení vnějšího počítačového systému 310 představuje minimální riziko pro vnější zabezpečenost a žádným způsobem nevystavuje vnitřní síť.

S použitím tohoto konkrétního provedení mohou existující http servery implementovat SGI s malým množstvím úprav nebo s žádnými existujícími příkazy cgi-bin. SGI je úplně skryto a bude automaticky podporovat dokonce i složité http servery. Je možné přidat dodatečné zabezpečení a podporu pro obchodní transakce s malými úpravami existujícího http serveru. Protože transakce (programy jako example.pl) dostupné vnější síti jsou omezeny na perl skripty a transakční programy se nacházejí v cgi-bin 415 a 426 v uvedeném pořadí a protože vnější server 320 by byl normálně pod přísnou kontrolou podniku a nebyl by snadno upravitelný vnitřními vývojovými pracovníky, ztěžuje SGI také vnitřním vývojovým pracovníkům zpřístupnění provádění vnitřních transakcí vnějším zákazníkům bez podnikového sledování a schvalování.

V obměně tohoto vynálezu spojuje alternativní provedení funkčnost rutiny 416 sgi_client a vnějšího démona do démona 314. To by zajistilo vyšší výkonnost, ale způsobilo by nutnost vlastnit http implementaci a její vylepšení by se těžko přidávala.

PATENTOVÉ NÁROKY

1. Způsob řízení vnitřního počítačového systému (320) pro umožnění vnějšímu počítačovému systému (310) zahájit požadavek (410) na transakci pomocí vnitřních prostředků (330) bez narušení bezpečnostního firewallu (300) mezi vnitřním počítačovým systémem a vnějším počítačovým systémem (310), **vyznačující se tím**, že:

se ověří (460) spojení zahájené vnitřním počítačovým systémem (320) mezi vnitřním počítačovým systémem (320) a vnějším počítačovým systémem (310) a tím se založí ověřené spojení;

vnější počítačový systém (310) zavolá (461) požadavek (410) na transakci přijatý vnějším počítačovým systémem (310);

odezvou na volání požadavku (410) na transakci, vytvoří (462) vnější počítačový systém (310) původní prostředí procesu obsahujícího proměnné prostředí procesu a vytvoří řetězec obsahující požadavek (410) na transakci a proměnné prostředí procesu pro spuštění požadavku (410) na transakci;

vnější počítačový systém (310) vyšle (435, 436, 437) řetězec do vnitřního počítačového systému (320) ověřeným spojením;

vnitřní počítačový systém (320) ověří (454) požadavek (410) na transakci;

vnitřní počítačový systém (320) obnoví (471) původní prostředí procesu; a

vnitřní počítačový systém (320) spustí (472) požadavek (410) na transakci a tím generování výstupu.

2. Způsob podle nároku 1, **vyznačující se tím**, že dále obsahuje následující kroky:

(a) vnější počítačový systém (310) přečte první heslo a první komunikační port;

(b) vnější počítačový systém (310) vytvoří první soket na prvním komunikačním portu a poslouchá první soket na volání spojení od vnitřního počítačového systému (320);

(c) vnitřní počítačový systém (320) přečte druhé heslo a druhý komunikační port; a

(d) vnitřní počítačový systém (320) vytvoří druhý soket na druhém komunikačním portu a pošle volání spojení na vnější počítačový systém (310) druhým soketem a tím založí spojení.

3. Způsob podle nároku 2, **vyznačující se tím**, že během ověřování (460)

(e) vnitřní počítačový systém (320) pošle jedinečnou časovou značku vnějšímu počítačovému systému (310) druhým soketem;

(f) vnější počítačový systém (310) upraví první heslo pomocí získané časové značky;

(g) upravené první heslo se zašifruje šifrovacím algoritmem a tak se vytvoří první řetězec hesla;

(h) vnější počítačový systém (310) vyšle vnitřnímu počítačovému systému (320) první řetězec hesla;

(i) vnitřní počítačový systém (320) opakuje kroky (f) až (g) s použitím druhého hesla a vytváří tak druhý řetězec hesla; a

(j) vnitřní počítačový systém (320) porovnává první řetězec hesla s druhým řetězcem hesla.

4. Způsob podle nároku 3, **vyznačující se tím**, že během úpravy

se provede logická operace AND časové značky s hexadecimálním číslem 0177; a

s výsledkem této logické operace AND se provede logická operace OR s každým znakem prvního hesla a tím se vytvoří upravené první heslo.

5. Způsob podle nároku 3, **vyznačující se tím**, že během šifrování se každý znak upraveného druhého hesla zašifruje klíčem.

6. Způsob podle nároku 1, **vyznačující se tím**, že během volání (461)

vnější síť pošle požadavek (410) na transakci vnějšímu počítačovému systému (310), kde požadavek (410) na transakci obsahuje vstupní data, parametry a příkaz pro spuštění transakčního programu; a

přijme se požadavek (410) na transakci vnějším počítačovým systémem (310), přičemž odezvou na něj první démon (314) definuje prostředí procesu obsahující proměnné prostředí procesu.

7. Způsob podle nároku 6, **vyznačující se tím**, že se odezvou na volání požadavku (410) na transakci vnějším počítačovým systémem (310) zavolá příkaz;

odezvou na zavolání příkazu se spustí skript, přičemž se mu pošlou uživatelská vstupní data, parametry a požadavek (410) na transakci; a

skript vytvoří (462) řetězec, kde řetězec obsahuje příkaz, argumenty a proměnné prostředí procesu pro spuštění požadavku (410) na transakci.

8. Způsob podle nároku 7, **vyznačující se tím**, že skript zavolá (463) klientskou rutinu (416) nacházející se ve vnějším počítačovém systému (310), předá jí uživatelská vstupní data, třetí komunikační port pro připojení ke druhému démonu (312), který se nachází ve vnějším počítačovém systému, (310) a identifikátor, který identifikuje typ požadavku (410) na transakci, a řetězec.

9. Způsob podle nároku 8, **vyznačující se tím**, že dále obsahuje následující kroky:

odezvou na přijetí volání skriptem druhý démon (312) ověří klientskou rutinu (416);

druhý démon (312) se rozvětví (453), přičemž se předá třetí soketové spojení odvozenému procesu a rodičovský proces poslouchá na prvním soketovém spojení na volání od vnitřního počítačového systému (320); a

odezvou na ověření klientské rutiny (416) odvozený proces vyšle typ požadavku (410) na transakci třetímu démonu (322), který se nachází ve vnitřním počítačovém systému (320).

10. Způsob podle nároku 9, **vyznačující se tím**, že během ověřování

třetí démon (322) čte tabulku platných služeb uloženou v paměti ve vnějším počítačovém systému (310); a

typ požadavku (410) na transakci přijatý od odvozeného procesu se porovná (454) s tabulkou platných služeb, kde pokud se typ v tabulce platných služeb nalezne, požadavek na transakci se ověří.

11. Systém pro řízení počítačového systému obsahujícího vnitřní počítačový systém (320) a vnější počítačový systém (310), **vyznačující se tím**, že

vnější počítačový systém (310) obsahuje:

prvního démona (314) pro zavolání požadavku (410) na transakci přijatého vnějším počítačovým systémem (310);

prostředky (415) pro vytvoření původního prostředí procesu obsahujícího proměnné prostředí procesu odezvou na zavolání požadavku (410) na transakci, a prostředky pro vytvoření řetězce obsahujícího požadavek (410) na transakci, argumenty a proměnné prostředí procesu pro spuštění požadavku (410) na transakci; a

druhého démona (312) pro vyslání řetězce vnitřnímu počítačovému systému (320) ověřeným spojením;

a vnitřní počítačový systém (320) obsahuje:

třetího démona (322) pro ověření spojení zahájeného vnitřním počítačovým systémem (320) mezi vnitřním počítačovým systémem (320) a vnějším počítačovým systémem (310) a tím zahájení ověřeného spojení; a pro ověření požadavku (410) na transakci;

čtvrtého démona (324) pro obnovení původního prostředí procesu a pro spuštění požadavku (410) na transakci a tím generování výstupu.

12. Systém podle nároku 11, **vyznačující se tím**, že

vnější počítačový systém (310) obsahuje:

počítačem čitelné prostředky programového kódu pro volání požadavku (410) na transakci;

počítačem čitelné prostředky programového kódu pro vytvoření původního prostředí procesu obsahujícího proměnné prostředí procesu, a počítačem čitelné prostředky programového kódu pro vytvoření řetězce obsahujícího požadavek (410) na transakci a proměnné prostředí procesu;

počítačem čitelné prostředky programového kódu pro vyslání řetězce vnitřnímu počítačovému systému (320);

a vnitřní počítačový systém (320) obsahuje:

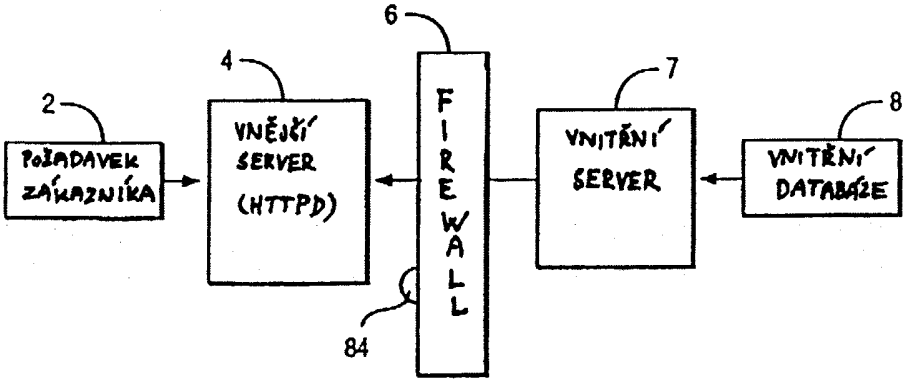
počítačem čitelné prostředky programového kódu pro ověření spojení mezi vnitřním počítačovým systémem (320) a vnějším počítačovým systémem (310);

počítačem čitelné prostředky programového kódu pro ověření požadavku (410) na transakci;

počítačem čitelné prostředky programového kódu pro obnovení původního prostředí procesu; a

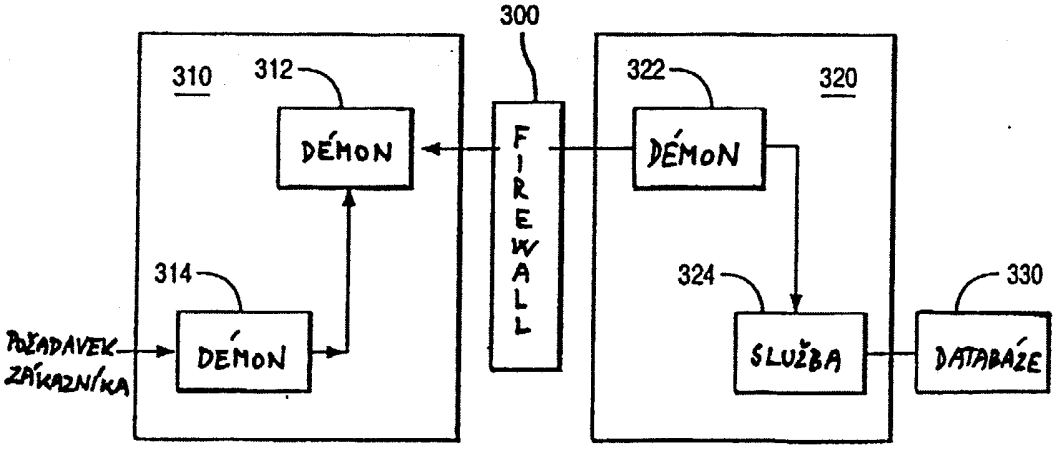
počítačem čitelné prostředky programového kódu pro spuštění požadavku (410) na transakci a tím generování výstupu.

4 výkresy

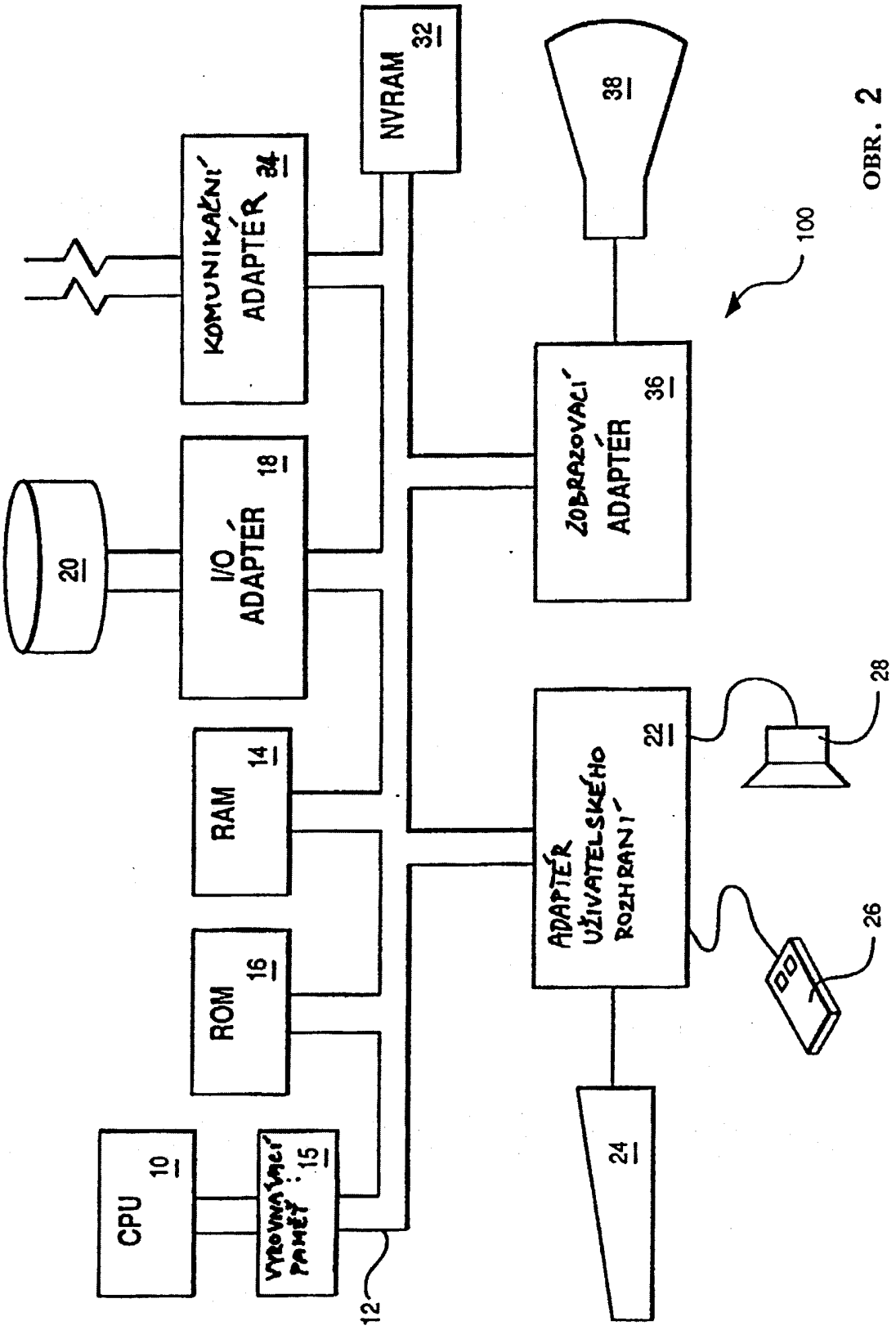


OBR. 1

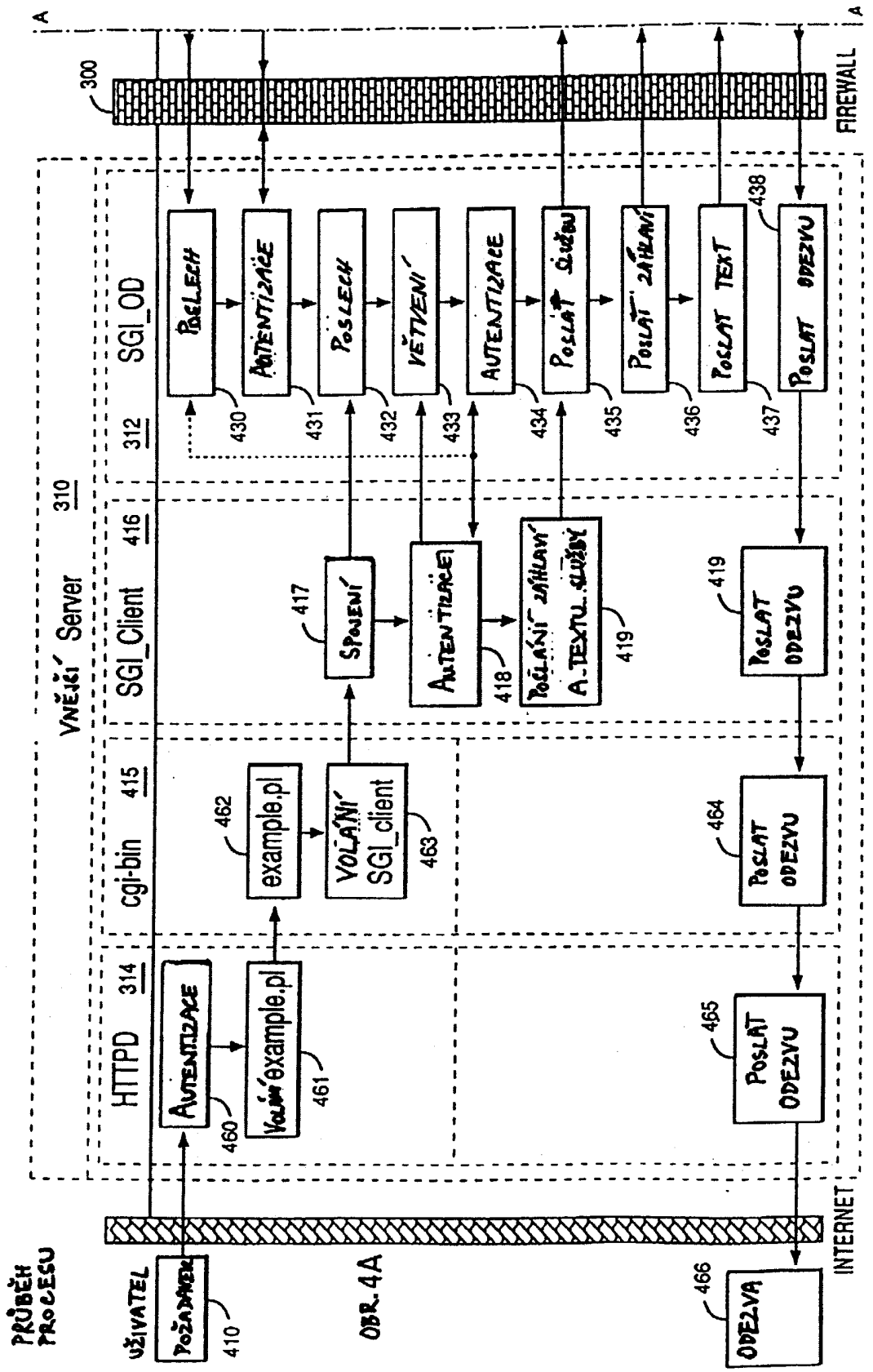
PŘEDCHOZÍ STAV TECHNIKY



OBR. 3

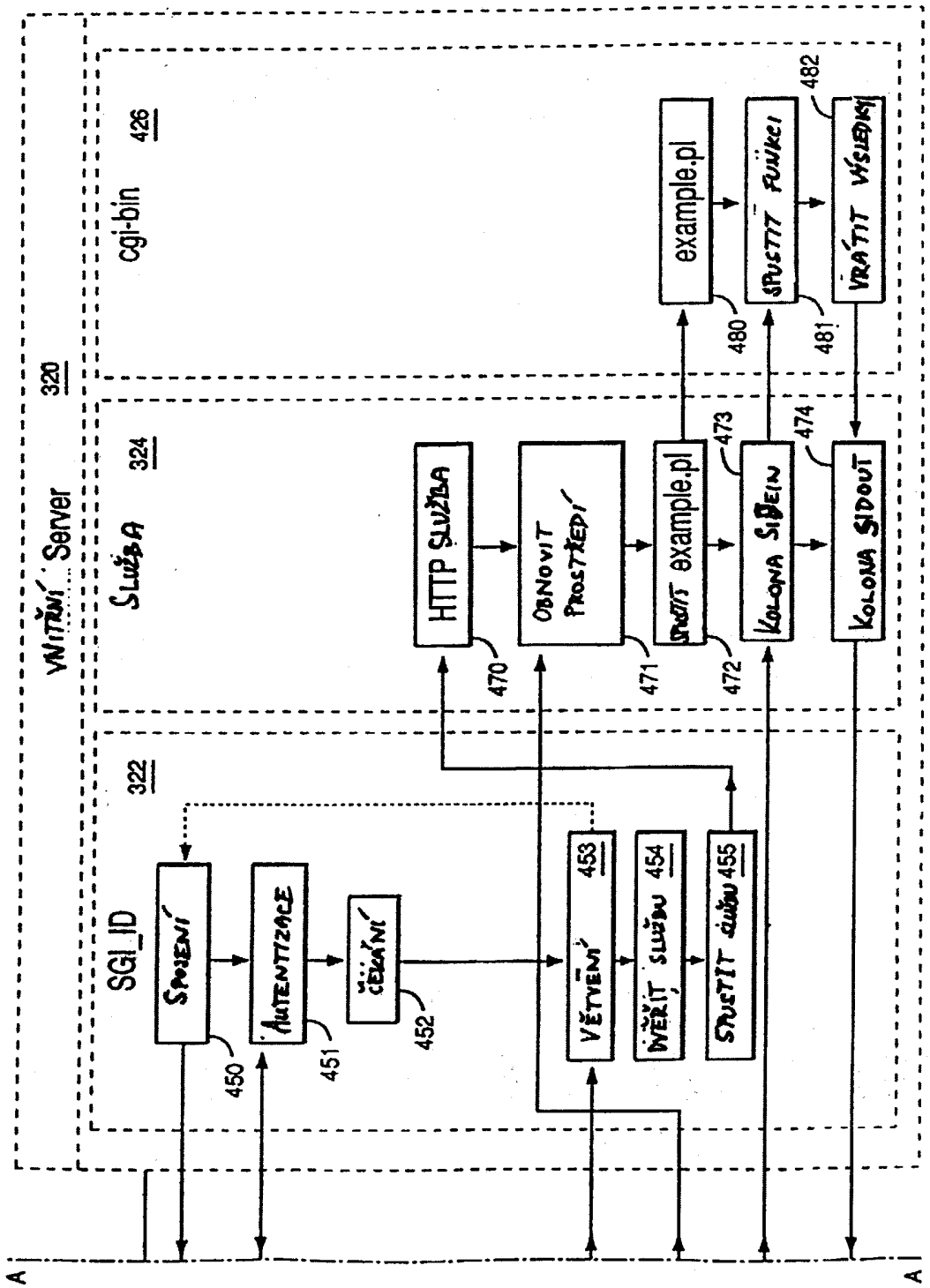

NVRAM 32
38
KOMUNIKAČNÍ ADAPTÉR 34
100
ZOBRAZOVACÍ ADAPTÉR 36
20
I/O ADAPTÉR 18
RAM 14
ROM 16
ADAPTÉR UŽIVATELSKÉHO ROZHRANÍ 22
28
26
CPU 10
VYROVNÁVACÍ PAMĚŤ 15
12
24

OBR. 2


PRŮBĚH PROCESU
UŽIVATEL
POŽADAVEK
410
INTERNET
ODEZVA
466
VNĚJŠÍ Server
310
HTTPD
314
AUTENTIZACE
460
VOLÁNÍ example.pl
461
cgi-bin
415
example.pl
462
VOLÁNÍ SGI_client
463
SGI_Client
416
SPOJENÍ
417
AUTENTIZACE
418
POSLÁNÍ ZÁHLAVÍ A TEXTU SLUŽBY
419
POSLAT ODEZVU
419
POSLAT ODEZVU
464
POSLAT ODEZVU
465
SGI_OD
312
POSLECH
430
AUTENTIZACE
431
POSLECH
432
VĚTVENÍ
433
AUTENTIZACE
434
POSLAT SLUŽBU
435
POSLAT ZÁHLAVÍ
436
POSLAT TEXT
437
POSLAT ODEZVU
438
300
FIREWALL
A
A

OBR. 4A

OBR. 4B



Konec dokumentu