

GITHUB AUTENTIFIKACIJA

Prevedeni tekstovi sa GitHub-a

SADRŽAJ

Obradena autentifikacija pomoću GitHuba i Gita, pristupi ključevima, podešavanje agenta, generiranje novih ključeva, dodavanje ključeva, testiranje SSH veze i rad s frazama lozinki SSH ključa.

Branko Čelap

[Course title]

Autentifikacija GitHub-a	2
Objašnjenje SSH	2
Autentifikacija uobičajene lozinke	2
Pristup javnom ključu	3
Pristup javnom ključu s podrškom agentu	5
Pristup javnom ključu s proslijeđivanjem agentu	6
Kako funkcioniraju izazovi s ključevima (engl. Key Challenges)	9
Sigurnosni problemi s agentima s ključevima (engl. Key Agents)	9
Podešavanje SSH agenta za proslijeđivanje	10
Podešavanje SSH agenta za proslijeđivanje	11
Testiranje SSH agenta s proslijeđivanjem	11
Rješavanje problema SSH agenta s proslijeđivanjem	12
Upravljanje ključevima za implementaciju	14
SSH agent za proslijeđivanje	14
HTTPS kloniranje s OAuth tokenima	15
Implementacija ključeva.....	15
Pristupni tokeni za instalaciju GitHub aplikacije	17
Strojni korisnici.....	18
Provjera postojećih SSH ključeva	19
O SSH ključevima.....	19
Provjera postojećih SSH ključeva	19
Generiranje novog SSH ključa i njegovo dodavanje u ssh-agent	20
SSH šifre ključeva.....	20
Generiranje novog SSH ključa	20
Dodavanje vašeg SSH ključa ssh-agentu	21
Generiranje novog SSH ključa za hardverski sigurnosni ključ	21
Dodavanje novog SSH ključa vašem GitHub računu	22
O dodavanju SSH ključeva vašem računu.....	22
Dodavanje novog SSH ključa vašem računu.....	23
Testiranje vaše SSH veze	24
Rad sa frazama lozinki SSH ključa.....	25
O frazama lozinki za SSH ključeve	25

Autentifikacija GitHub-a

Objašnjenje SSH




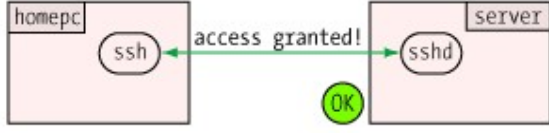
Autentifikacija uobičajene lozinke

SSH podržava pristup s korisničkim imenom i lozinkom, a ovo je nešto malo više od šifriranog telnet. Pristup je zapravo kao telnet, s normalnom razmjenom korisničkog imena/lozinke.

Koristeći SSH protokol, možete se povezati i autentificirati na udaljene servere i usluge. Pomoću SSH ključeva možete se povezati s npr. GitHubom bez unosa korisničkog imena i osobnog pristupnog tokena pri svakom posjetu. Također možete koristiti SSH ključ za potpisivanje *commitova*.

Primijetiti ćemo da ova razmjena, kao i sve ostale u ovom radu, pretpostavljaju da je početna razmjena **ključeva hosta** (centralnog kompjutera) uspješno dovršena. Iako je važan dio sigurnosti sesije, provjera validacije hosta nije bitna za raspravu o prosljeđivanju ključa agenta.

Svi primjeri počinju od korisnika na **homepc** (možda Windows radnoj stanici) koji se povezuje s PuTTY-em na server koji ima pokrenut OpenSSH. Pojedini detalji (uglavnom nazivi programa) razlikuju se od implementacije do implementacije, ali dokazano je da je temeljni protokol vrlo interoperabilan (dakle vrlo dobro međusobno funkcionira).

1.	Korisnik uspostavlja početnu TCP vezu i šalje korisničko ime. Primijetiti ćemo da za razliku od telnet, gdje se korisničko ime traži kao dio povezanog toka podataka (bez semantičkog značenja koje sam telnet razumije), razmjena korisničkog imena je dio samog ssh protokola.	
2.	ssh demon ¹ na serveru odgovara zahtjevom za lozinkom, a pristup sistemu još nije dopušten ni na koji način.	
3.	ssh klijent traži od korisnika lozinku, koja se kroz šifriranu veze prenosi na server gdje se uspoređuje s lokalnom korisničkom bazom.	
4.	Ako se korisnička lozinka podudara s lokalnim akreditivom (engl. credential), odobrava se pristup sistemu i uspostavlja se dvosmjerna komunikacijska staza, obično do ljuške za prijavu (engl. login shell).	

Glavna prednost dopuštanja provjere autentičnosti lozinkom jest to što ju je jednostavno postaviti — obično je zadana — i lako ju je razumjeti. Sistemi koji zahtijevaju pristup za mnogo korisnika s mnogo

¹ U multitasking operativnim sistemima, demon je kompjuterski program koji radi kao pozadinski proces, umjesto da bude pod direktnom kontrolom interaktivnog korisnika. Tradicionalno, imena procesa demona završavaju slovom **d**, radi pojašnjenja da je proces zapravo demon i radi razlikovanja između demona i normalnog računalnog programa. Na primjer, syslogd je demon koji implementira mogućnost zapisivanja sistema, a sshd je demon koji posluhuje dolazne SSH veze.

različitih lokacija često dopuštaju autentifikaciju lozinke jednostavno kako bi se smanjio administrativni teret i maksimizirao pristup.

Značajna mana je da dopuštanjem **korisniku** da unese lozinku, to znači da **svatko** može unijeti lozinku. Ovo otvara vrata masovnom pogađanju lozinke od strane korisnika ili botova, a to je sve češća metoda kompromitacije sistema.

Za razliku od prethodne generacije ssh crva, koji su pokušali unijeti samo nekoliko uobičajenih lozinki s uobičajenim korisničkim imenima, napredan zloćudni softver ima vrlo opsežan rječnik korisničkih imena i lozinki i pokazao se najučinkovitijim u probijanju čak i u sisteme s "dobrim" lozinkama. Za ulazak u sistem potreban je samo jedan kompromitiran račun.

Ali čak i ako ostavimo sigurnosna pitanja po strani, drugi nedostatak provjere autentičnosti lozinke je da se lozinke moraju zapamtiti i unijeti odvojeno nakon svake prijave. Za korisnike sa samo jednim sistemom za pristup, ovo možda neće biti toliko opterećenje, ali korisnicima koji se povezuju na mnogo sistema tokom dana može biti zamorno ponavljanje unosa lozinke.

A pamtiti različite lozinke za svaki sistem nije pogodno za odabir jakih lozinki.

Za:	jednostavan za postavljanje
Protiv:	omogućava brutalno pogađanje lozinke
Protiv:	svaki puta zahtijeva pogađanje lozinke

Pristup javnom ključu

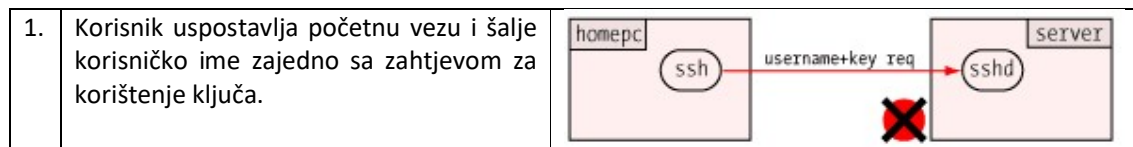
Kako bi se suprotstavio nedostacima provjere autentičnosti lozinke, ssh podržava pristup javnom ključu. Korisnik stvara par javnih i privatnih ključeva i instalira javni ključ u svoju datoteku **\$HOME/.ssh/authorized_keys** na ciljnom serveru. Ovo su neosjetljive informacije koje se ne moraju čuvati, ali druga polovica - privatni ključ - zaštićena je na lokalnom kompjuteru (nadajmo se) jakim frazom lozinke (engl. passphrase).



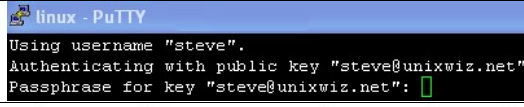
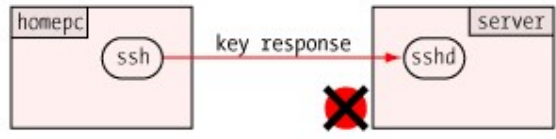

Javni ključ je dugačak string bitova kodiranih u ASCII i pohranjen je u jednom dugom redu (iako je ovdje predstavljen u tri reda koja se nastavljaju radi čitljivosti). Uključuje tip (**ssh-rsa** ili drugi), sam ključ i komentar:

`$HOME/.ssh/authorized_keys`

```
ssh-rsa AzAAB3NzaC1yc2EaaaaabiWaaaieaX9AyNR7xWnW0eI3x2NGXrJ4gkQpK/EqpkveGCvvbM \
oH84zqu3Us8jSaQD392JZAEAhGSoe0dWMBFm9Y41VGZYMncwkftQPFH1P07vDw49aTAa2RJNFyV \
QANZCbSocDeuT0Q7usuUj/v8h27+PqsUU79XVQSDIhXBkW+bJawc1c= Steve's key
```

Ovaj ključ mora biti instaliran na ciljnom sistemu — jednokratno — gdje se koristi za kasniji daljinski pristup vlasnika privatnog ključa.



2.	ssh demon na serveru traži korisničku datoteku authorized_keys , konstruira izazov (engl. challenge) na osnovi javnog ključa koji se tamo nalazi i šalje ovaj izazov natrag korisnikovom ssh klijentu.	
3.	Ssh klijent prima izazov s ključem (engl. key challenge). Pronalazi korisnički privatni ključ na lokalnom sistemu, ali je zaštićen frazom lozinke. Datoteka RSA ključa naziva se id_rsa na OpenSSH i SecureCRT , keyname.ppk na PuTTY . Drugi tipovi ključeva (DSA, na primjer) imaju slične formate naziva.	
4.	Od korisnika se traži fraza lozinke za otključavanje privatnog ključa. Ovaj primjer je iz PuTTY-ja .	
5.	ssh koristi privatni ključ za konstruiranje odgovora s ključem (engl. key response) i šalje ga sshd-u koji čeka na drugom kraju veze. Ne šalje sam privatni ključ!	
6.	sshd provjerava odgovor ključa i ako je validan, odobrava pristup sistemu.	

Ovaj proces uključuje više koraka iza događaja ali korisničko iskustvo uglavnom je isto: od vas se traži **frazu lozinke** umjesto **lozinke**. No, za razliku od postavljanja pristupa višestrukim računalnim sistemima (od kojih svaki može imati različitu lozinku), korištenje pristupa javnim ključem znači da upisujete istu frazu lozinke bez obzira na koji se sistem povezujete.

To ima značajan ali neočit sigurnosni benefit: budući da ste sada odgovorni za samo jednu tajnu frazu umjesto za puno lozinke, upisujete je češće. Zbog toga je veća vjerojatnost da ćete je zapamtiti i zato odabrati jaču šifru za zaštitu privatnog ključa nego što biste inače mogli.

Pokušaj pamćenja puno odvojenih lozinke za različite udaljene sisteme je teško i nije moguće za jake lozinke. Pristup javnim ključem u potpunosti rješava ovaj problem.

Napomenut ćemo da, iako se računi s javnim ključem općenito ne mogu probiti na daljinu, sama instalacija javnog ključa na ciljnom sistemu ne onemogućuje korištenje lozinke u cijelom sistemu. Umjesto toga, server mora biti eksplicitno konfiguriran da dopušta samo enkripciju javnim ključem korištenjem ključnih riječi **PasswordAuthentication no** u **sshd_config** datoteci.

Za:	javni ključevi se ne mogu lako probiti (engl. brute-forced) na silu
Za:	isti privatni ključ (sa frazom lozinke) može se koristiti za pristup više sistema: nema potrebe pamtiti mnogo lozinke
Protiv:	zahtijeva jednokratno postavljanje javnog ključa na ciljnom sistemu
Protiv:	zahtijeva otključavanje privatnog ključa s tajnom frazom lozinke nakon svake konekcije

Pristup javnom ključu s podrškom agentu

Sada kada smo napravili korak u pristupu javnom ključu, poduzet ćemo sljedeći korak kako bismo omogućili podršku za agente. U prethodnom odlomku, korisnički privatni ključ bio je otključan kod svakog zahtjeva za povezivanjem: ovo se funkcionalno ne razlikuje od upisivanja lozinke, i iako je svaki put ista fraza lozinke (što je čini uobičajnom), svejedno postaje zamorno na isti način.

Srećom, ssh paket nudi posrednika poznatog kao "key agent" koji može držati i upravljati privatnim ključevima na vašim radnim stanicama i odgovarati na zahtjeve udaljenih sistema za provjeru vaših ključeva. Agenti pružaju ogromnu korist u produktivnosti jer nakon što otključate svoj privatni ključ (jednom kada pokrenete agenta), kasniji pristup radi s agentom bez upita.

Ovo funkcionira kao prethodno viđenog pristupa s ključem, ali s dodatkom.

1.	Korisnik uspostavlja početnu vezu i šalje korisničko ime zajedno sa zahtjevom za korištenje ključa.	
2.	ssh demon na serveru gleda [1] u korisničku datoteku <code>authorized_keys</code> , konstruira izazov na temelju ključa i šalje ga [2] natrag korisnikovom ssh klijentu.	
3.	Ssh klijent prima izazov s ključem (engl. key challenge) i proslijeđuje ga agentu na čekanju. Agent, umjesto samog ssh-a, otvara privatni ključ korisnika i otkriva da je zaštićen frazom lozinke.	
4.	Od korisnika se traži fraza lozinke za otključavanje privatnog ključa. Ovaj primjer prikazuje upit iz PuTTY-a.	
5.	Agent konstruira odgovor ključa i vraća ga [1] ssh procesu, koji ga šalje [2] sshd- u koji čeka na drugom kraju. Za razliku od prethodnog primjera, ssh nikada ne vidi privatni ključ direktno, samo odgovor ključa.	
6.	sshd provjerava odgovor ključa i ako je validan, odobrava pristup sistemu. Napomena: agent još uvijek zadržava privatne ključeve u memoriji, iako ne sudjeluje u razgovoru koji je u toku.	

Što se korisnika tiče, ova prva razmjena malo se razlikuje od pristupa ključu prikazanog u prethodnom odlomku: jedina razlika je koji program traži privatni ključ (sam `ssh` nasuprot agenta).

Ali podrška agenta blista je kod sljedeće zahtjeva za konekcijom dok je agent još uvijek prisutan. Budući da pamti privatne ključeve od prvog puta kada je otključan pomoću fraze lozinke, može odmah odgovoriti na izazov ključa **bez upita**. Korisnik vidi trenutnu, direktnu prijavu bez potrebe da bilo što upisuje.

Mnogi korisnici otključaju svoje privatne ključeve samo jednom ujutro kada pokreću svoj ssh klijent i agenta, i ne moraju ga ponovno unositi ostatak dana jer rezidentan agent rukuje svim izazovima s ključem. Nevjerojatno je praktičan, ali i siguran.

Vrlo je važno razumjeti da privatni ključevi **nikada ne napuštaju agenta**: umjesto toga, klijenti traže od agenta da izvrši računanje na temelju ključa, a to se radi na način koji omogućuje agentu da dokaže da ima privatni ključ bez potrebe otkriti sam ključ. O izazovu/odgovoru raspravljamo u kasnijem odlomku.

Jednom kada je podrška za agente omogućena, svi upiti su sada zaobiđeni i može se razmotriti izvođenje skriptiranih ažuriranja udaljenih sistema. Ovaj izmišljeni primjer kopira konfiguracijsku datoteku za prijavu `.bashrc` na svaki udaljeni sistem, a zatim provjerava koliko se diskovnog prostora koristi (pomoću naredbe `df`):

```
# skriptirano ažuriranje nekoliko udaljenih sistema
for svr in server1 server2 server3 server4
do
    scp .bashrc $svr:~/ # kopira novi .bashrc
    ssh $svr df         # pita za prostor na disku
done
```

Bez podrške agenta, svaki bi server **zahtijevao dva upita** (prvo za kopiranje, a zatim za daljinsko izvršenje naredbe). Uz podršku agenta, uopće nema upita.

Međutim, ove pogodnosti se prikupljaju samo za odlazne veze (engl. outbound connections) napravljene iz lokalnog sistema na ssh servere negdje drugdje: jednom kada se prijavite na udaljeni server, povezivanje s njega na neki treći server zahtijeva ili pristup lozinkom ili postavljanje privatnog ključa korisnika na međusistemu prelaska u treći.

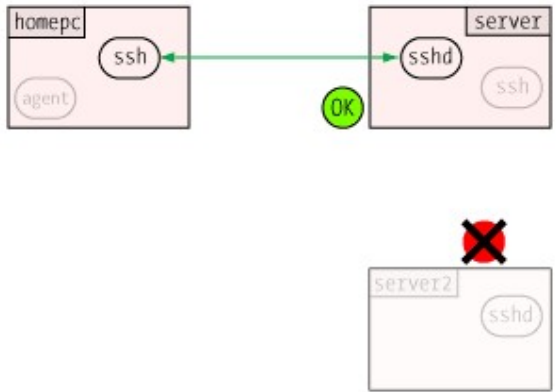
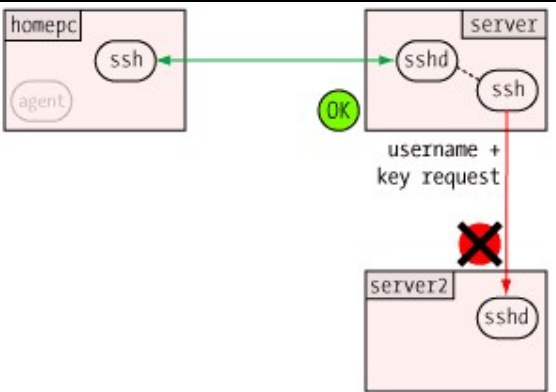
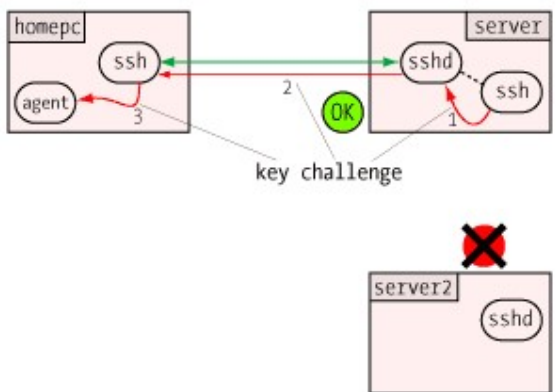
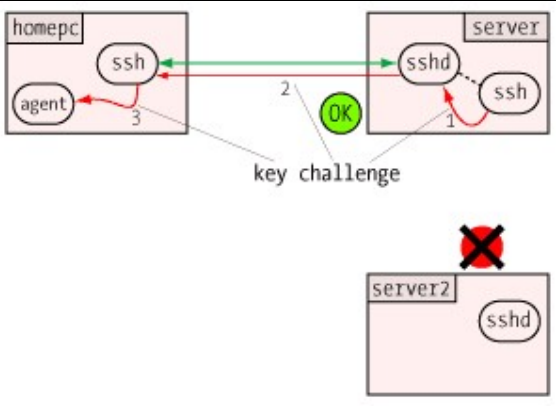
Imati podršku agenta na lokalnom sistemu svakako je poboljšanje, ali mnogi od nas koji rade na daljinu često moraju kopirati datoteke s jednog udaljenog sistema na drugi. Bez instaliranja i pokretanja agenta na prvom udaljenom sistemu, **scp** operacija će svaki put zahtijevati lozinku ili frazu lozinke. U određenom smislu, ovo samo gura izazov nazad jednu kariku u ssh lanacu.

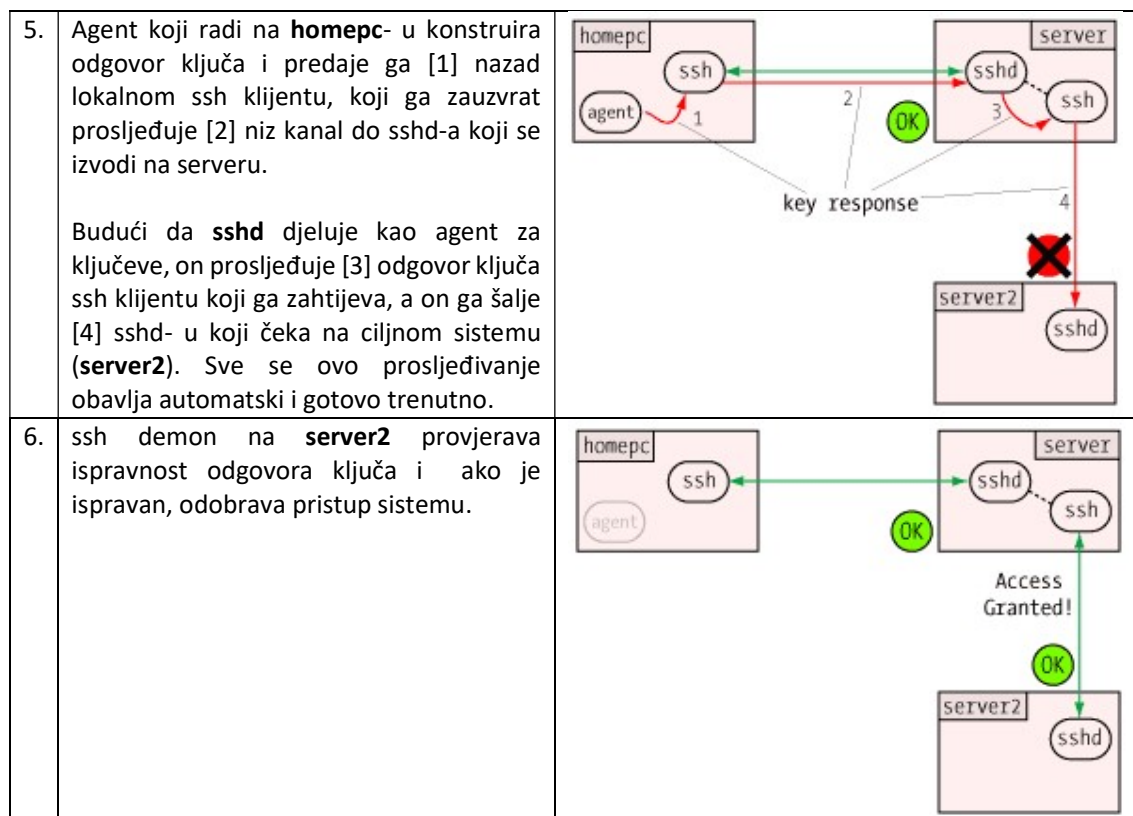
Srećom, postoji odlično rješenje koje rješava sve te probleme.

Za:	Zahtijeva otključavanje privatnog ključa samo jednom
Za:	Olakšava skriptirani daljinski rad na više sistema
Protiv:	Jednokratni trošak za podešavanje agenta
Protiv:	Zahtijeva privatni ključ na udaljenim klijentskim računalima ako žele uspostaviti daljnje izlazne konekcije

Pristup javnom ključu s proslijeđivanjem agentu

S našim agentom s ključevima na mjestu, vrijeme je da omogućimo posljednji dio naše slagalice: proslijeđivanje agenta (engl. agent forwarding). Ukratko, ovo omogućuje lancu ssh konekcija da proslijedi izazove s ključem natrag originalnom agentu, izbjegavajući potrebu za lozinkama ili privatnim ključevima na svim posrednim strojevima.

<p>1. Ovo sve počinje već uspostavljenom vezom s prvim serverom, pri čemu agent sada drži privatni ključ korisnika. Drugi server još ne igra nikakvu ulogu.</p>	
<p>2. Korisnik pokreće ssh klijent na prvom serveru sa zahtjevom za povezivanje s server2, a to proslijeđuje korisničko ime i zahtjev za korištenjem ključa ssh demonu (ovo se također može učiniti s scp secure copy naredbom)</p>	
<p>3. Ssh demon pregledava korisničku datoteku authorized_keys [1], konstruira izazov za ključ iz ključa i šalje ga [2] natrag niz kanal do klijenta koji je napravio zahtjev.</p>	
<p>4. Ovdje se događa magija: ssh klijent na serveru prima izazov za ključ od ciljnog sistema i proslijeđuje [1] taj izazov sshd serveru na istom stroju koji djeluje kao agent za ključeve (engl. „key agent“).</p> <p>sshd zauzvrat prenosi [2] izazov za ključ niz prvu konekciju na originalni ssh klijent. Kada se vrati na homepc, ssh klijent poduzima posljednji korak u procesu prijenosa rukovanjem izazova ključa [3] rezidentnom agentu, koji zna za privatni ključ korisnika.</p>	



Ovaj se proces može ponoviti s još više veza u lancu (recimo, ako je korisnik želio ssh s **server2** na **server3**), i sve se događa automatski. Podržava cijeli paket programa povezanih sa ssh-om, kao što su **ssh**, **scp** (sigurna kopija) i **sftp** (sigurni prijenos datoteka poput FTP-a).

To zahtijeva jednokratnu instalaciju korisnikove javne — **ne privatne!** — ključeve na svim ciljanim strojevima, ali ovaj trošak postavljanja se brzo nadoknađuje dodatnom produktivnošću. Oni koji koriste javne ključeve s proslijeđivanjem agenta rijetko se vraćaju na nešto drugo.

Za:	Izuzetna praktičnost
Protiv:	istemima
Protiv:	Za razumijevanje je potreban tehnički savjet
Za:	Dostupan je izvrstan tehnički savjet :-)

Kako funkcioniraju izazovi s ključevima (engl. Key Challenges)

Jedan od pametnijih aspekata agenta je način na koji može provjeriti identitet korisnika (ili točnije, posjedovanje privatnog ključa) bez da taj privatni ključ ikome otkrije. Ovo, kao i mnoge druge stvari u modernim sigurnim komunikacijama, koristi enkripciju s javnim ključem.

Kada korisnik želi pristup ssh serveru, on predstavlja svoje korisničko ime serveru sa zahtjevom za postavljanje ključne sesije. Ovo korisničko ime pomaže pronaći popis javnih ključeva kojima je dopušten pristup tom serveru: obično se nalazi u datoteci `$HOME/.ssh/authorized_keys`.

Server stvara "izazov" (engl. challenge) na koji može odgovoriti samo

onaj tko posjeduje odgovarajući privatni ključ: stvara i pamti veliki slučajni broj, zatim ga šifrira s korisničkim javnim ključem. Ovo stvara međuspremnik binarnih podataka koji se šalje korisniku koji zahtijeva pristup. Za bilo koga bez privatnog ključa, to je samo hrpa bitova.

Kada agent primi izazov, dešifrira ga privatnim ključem. Ako je ovaj ključ "druga polovina" javnog ključa na serveru, dešifriranje će biti uspješno, otkrivajući originalni slučajni broj koji je generirao server. Samo vlasnik privatnog ključa je ikada mogao izdvojiti ovaj slučajni broj, tako da to predstavlja dokaz da je korisnik vlasnik privatnog ključa.

Agent uzima ovaj slučajni broj, dodaje SSH sesiji ID (koji se razlikuje od konekcije do konekcije) i kreira MD5 hash vrijednost rezultirajućeg stringa: ovaj se rezultat šalje nazad na server kao odgovor za ključ (engl. key response).

Server izračunava isti MD5 hash (slučajni broj + ID sesije) i uspoređuje ga s odgovorom za ključ od agenta: ako se poklapaju, korisnik je morao posjedovati privatni ključ i

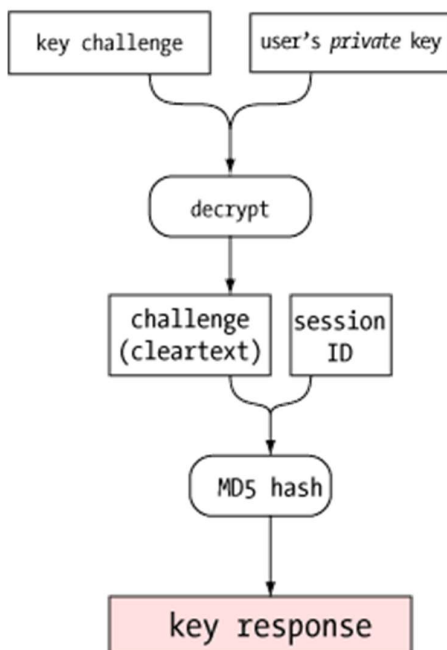
pristup je odobren. Ako nije, sljedeći ključ na popisu (bilo koji) se pokušava uzastopno dok se ne pronađe važeći ključ ili više nema dostupnih ovlaštenih ključeva. U tom trenutku pristup je odbijen.

Zanimljivo, stvarni slučajni broj nikada nije izložen u razmjeni klijent/agent - on se šifriran šalje agentu i uključuje u MD5 hash od agenta. Vjerojatno je ovo sigurnosna mjera opreza osmišljena da oteža karakterizaciju svojstava generatora slučajnih brojeva na serveru gledajući klijent/agent razmjenu.

Sigurnosni problemi s agentima s ključevima (engl. Key Agents)

Jedna od sigurnosnih prednosti prosljeđivanja agenta je ta da se privatni ključ korisnika nikada ne pojavljuje na udaljenim sustavima ili na žici, čak ni u šifriranom obliku. Ali isti protokol agenta koji štiti privatni ključ ipak može otkriti drugačiju ranjivost: otmicu agenta.

Svaka implementacija ssh-a mora osigurati mehanizam za klijente da zatraže usluge agenta, a na UNIX/Linuxu to se obično radi s utičnicom UNIX domene pohranjenom u direktoriju `/tmp/`. Na našem

Key Challenge Creation**Key Response Generation**

Linux sustavu koji izvodi OpenSSH, na primjer, nalazimo datoteku `/tmp/ssh-CXkd6094/agent.6094` povezanu sa SSH demonom koji servisira SecureCRT udaljenog klijenta.

Ova socket datoteka je zaštićena onoliko koliko operativni sustav dopušta (ograničena samo na korisnika koji pokreće proces, čuva se u zaštićenom poddirektoriju), ali ništa ne može spriječiti root korisnika da pristupi bilo kojoj datoteci bilo gdje.

Ako root korisnik može uvjeriti svog ssh klijenta da koristi agenta drugog korisnika, root može oponašati tog korisnika na bilo kojem udaljenom sustavu koji autorizira javni ključ žrtve. Naravno, root to može učiniti i na lokalnom sustavu, ali svejedno to može učiniti direktno bez potrebe za ssh trikovima.

Nekoliko varijabli okoline koristi se za usmjeravanje klijenta na agenta, ali samo je `SSH_AUTH_SOCK` potreban da bi se koristilo prosljeđivanje agenta. Postavljanje ove varijable na utičnicu agenta žrtve omogućuje potpunu upotrebu te utičnice ako je temeljna datoteka čitljiva. Za `root` uvijek jest.

```
# ls -l /tmp/ssh*      - potraži neki agent socket
/tmp/ssh-CXkd6094:
total 24
srwxr-xr-x  1 steve  steve          0 Aug 30 08:46 agent.6094=
# export SSH_AUTH_SOCK=/tmp/ssh-CXkd6094/agent.6094
# ssh steve@remotesystem
remote$              - Imam te! Prijavljen kao "steve" korisnik na udaljenom sistemu!
```

Ne može se reći koji će daljinski sistemi prihvatiti korisnički ključ samo na temelju informacija o `socket-u`, ali nije potrebno previše detektivskog rada da bi se ušlo u trag. Periodično pokretanje naredbe `ps` na lokalnom sistemu može pokazati da korisnik izvodi `ssh remotesystem`, a naredba `netstat` može upućivati na korisničku matičnu bazu.

Nadalje, korisnička datoteka `$HOME/.ssh/known_hosts` sadrži popis strojeva s kojima korisnik ima vezu: iako možda nisu svi konfigurirani da vjeruju korisničkom ključu, to je svakako odlično mjesto za početak traženja. Moderne verzije (4.0 i novije) OpenSSH-a mogu opcionalno isjeći u blokove i povezati (engl. hash) datoteku `known_hosts` kako bi to spriječile.

Ne postoji tehnička metoda koja bi spriječila root korisnika da otme socket SSH agenta ako ima mogućnost pristupa, tako da ovo sugerira da prosljeđivanje agenta možda i nije tako dobra ideja kada se udaljenom sistemu ne može u potpunosti vjerovati. Svi ssh klijenti pružaju metodu za onemogućavanje prosljeđivanja agenta.

Iako agent koristi socket, to je socket UNIX domene, kojem se može pristupiti samo iz lokalnog datotečnog sistema. Agenti ne slušaju nijedan TCP/IP socket

Podešavanje SSH agenta za prosljeđivanje

Prosljeđivanje SSH agenta može se koristiti za jednostavno dislociranje na server. Omogućava vam korištenje lokalnih SSH ključeva umjesto da ključeve (bez fraza lozinki!) ostavite na svom serveru.

Ako ste već podesili SSH ključ za interakciju s GitHubom, vjerojatno ste upoznati s `ssh-agent`. To je program koji radi u pozadini i čuva vaš ključ učitanim u memoriju, tako da ne morate unositi frazu lozinke svaki puta kada trebate upotrijebiti ključ. Zgodna stvar je što možete dopustiti serverima da pristupaju vašem lokalnom `ssh-agent` kao da je već pokrenuti na serveru. Ovo je kao da tražite od prijatelja da unese svoju lozinku kako biste mogli koristiti njegov kompjuter.

Podešavanje SSH agenta za prosljeđivanje

Provjerite je li vaš vlastiti SSH ključ postavljen i radi. Možete testirati da vaš lokalni ključ radi unosom `ssh -T git@github.com` terminal:

```
$ ssh -T git@github.com
# Attempt to SSH in to github
> Hi IMEKORISNIKA! You've successfully authenticated, but GitHub does not provide
shell access.
```

Postavimo SSH da dopustimo prosljeđivanje agenta na vaš server.

1. Koristeći svoj omiljeni editor teksta, otvorite datoteku na `~/.ssh/config`. Ako ova datoteka ne postoji, možete je stvoriti unosom `touch ~/.ssh/config` u terminal.
2. Unesite sljedeći tekst u datoteku, zamijenivši ga primjer.com imenom ili IP-om domene vašeg servera:

```
Host primjer.com
  ForwardAgent yes
```

Upozorenje: Možda ćete doći u iskušenje da upotrijebite zamjenski znak kao `Host *` da samo primijenite ovu postavku na sve SSH veze. To baš i nije dobra ideja, jer biste svoje lokalne SSH ključeve dijelili sa svakim serverom na koji se upustite SSH. Oni neće imati direktan pristup ključevima, ali će ih moći koristiti kao i vi dok je veza uspostavljena. Trebali biste dodati samo servere u koje vjerujete i koje namjeravate koristiti s prosljeđivanjem agenta.

Testiranje SSH agenta s prosljeđivanjem

Da biste provjerili funkcionira li agenta s prosljeđivanjem s vašim serverom, možete se upustiti u SSH na server i pokrenuti ga `ssh -T git@github.com` još jednom. Ako je sve u redu, dobit ćete isti odziv kao i lokalno.

Ako niste sigurni koristi li se vaš lokalni ključ, također možete provjeriti `SSH_AUTH_SOCK` varijablu na svom serveru:

```
$ echo "$SSH_AUTH_SOCK"
# Print out the SSH_AUTH_SOCK variable
> /tmp/ssh-4hNGMk8AZX/agent.79453
```

Ako varijabla nije podešena, to znači da agent s prosljeđivanjem ne radi:

```
$ echo "$SSH_AUTH_SOCK"
# Ispiši SSH_AUTH_SOCK variable
> [No output]
```

```
$ ssh -T git@github.com
# Pokušaj SSH na github-u
> Permission denied (publickey).
```

Rješavanje problema SSH agenta s prosljeđivanjem

Evo nekih stvari na koje treba obratiti pažnju kada rješavate probleme SSH agenta s prosljeđivanjem.

Morate koristiti SSH URL za provjeru koda

SSH prosljeđivanje radi samo sa SSH URL-ovima, ne i HTTP(s) URL-ovima. Provjerite `.git/config` datoteku na svom serveru i provjerite da li je URL u SSH stilu kao što je ispod:

```
[remote "origin"]
  url = git@github.com:VAŠ_RAČUN/VAŠ_PROJEKT.git
  fetch = +refs/heads/*:refs/remotes/origin/*
```

Vaši SSH ključevi moraju raditi lokalno

Prije nego što omogućite da vaši ključevi rade pomoću agenta s prosljeđivanjem agenta, oni prvo moraju raditi lokalno. Naš vodič za generiranje SSH ključeva može vam pomoći da postavite svoje SSH ključeve lokalno.

Vaš sistem mora omogućiti prosljeđivanje SSH agenta

Ponekad konfiguracije sustava onemogućavaju prosljeđivanje SSH agenta. Možete provjeriti koristi li se konfiguracijska datoteka sistema unosom sljedeće naredbe u terminal:

```
$ ssh -v URL
# Spoji se na navedeni URL s verbose debug output
> OpenSSH_8.1p1, LibreSSL 2.7.3
> debug1: Reading configuration data /Users/YOU/.ssh/config
> debug1: Applying options for example.com
> debug1: Reading configuration data /etc/ssh_config
> debug1: Applying options for *
$ exit
# Povratak na vaš lokalni komandni prompt
```

U primjeru iznad, datoteka `~/.ssh/config` se prvo učitava, a zatim `/etc/ssh_config` se čita. Tu datoteku možemo provjeriti da vidimo nadjačava li naše opcije pokretanjem sljedećih naredbi:

```
$ cat /etc/ssh_config
# Ispiši /etc/ssh_config datoteku
> Host *
>   SendEnv LANG LC_*
>   ForwardAgent no
```

U ovom primjeru, naša `/etc/ssh_config` datoteka posebno kaže `ForwardAgent no`, što je način za blokiranje agenta za prosljeđivanje. Brisanje ovog reda iz datoteke trebalo bi ponovno pokrenuti agenta za prosljeđivanje.

Vaš server mora dopustiti SSH agenta s prosljeđivanjem na ulaznim konekcijama

Prosljeđivanje agenta također može biti blokirano na vašem serveru. Možete provjeriti je da li je prosljeđivanje agenta dopušteno tako da SSH-om pristupite serveru i pokrenete `sshd_config`. Izlaz ove naredbe ukazuje da je `AllowAgentForwarding` postavljen.

Vaš lokal `ssh-agent` mora raditi

Na većini kompjutera, operativni sistem se automatski pokreće `ssh-agent` umjesto vas. Međutim, u Windows-ima to trebate učiniti ručno. Imamo vodič o tome kako započeti `ssh-agent` kad god otvorite Git Bash.

Da biste provjerili da li `ssh-agent` radi na vašem kompjuteru, unesite sljedeću naredbu u terminal:

```
$ echo "$SSH_AUTH_SOCK"
# Ispiši SSH_AUTH_SOCK varijablu
> /tmp/launch-kNSlgU/Listeners
```

Vaš ključ mora biti dostupan ssh-agent-u

Možete provjeriti da li je vaš ključ vidljiv `ssh-agent`-u pokretanjem sljedeće naredbe:

```
ssh-add -L
```

Ako naredba kaže da identitet nije dostupan, morat ćete dodati svoj ključ:

```
ssh-add VAŠ-KLJUČ
```

Na macOS-u `ssh-agent` će "zaboraviti" ovaj ključ nakon što ga ponovno pokrenete. Ali možete uvesti svoje SSH ključeve u Keychain (mjesto za pohranjivanje kriptografskih ključeva) pomoću ove naredbe:

```
ssh-add --apple-use-keychain VAŠ-KLJUČ
```

Napomena: `--apple-use-keychain` opcija pohranjuje frazu lozinke u vaš keychain za vas kada dodate SSH ključ `ssh-agentu`. Ako ste odabrali da svom ključu ne dodate frazu lozinke, pokrenite naredbu bez `--apple-use-keychain` opcije.

`--apple-use-keychain` opcija je u Appleovoj standardnoj verziji `ssh-add`. U verzijama macOS-a prije Montereyja (12.0), `--apple-use-keychain` i `--apple-load-keychain` koristile su sintaksu `-K` odnosno `-A`.

Ako nemate `ssh-add` instaliranu Appleovu standardnu verziju, možda ćete primiti grešku. Za više informacija pogledajte "Pogreška: ssh-add: ilegalna opcija -- apple-use-keychain ."

Ako se od vas i dalje traži vaša fraza lozinke, možda ćete morati dodati naredbu u svoju `~/.zshrc` datoteku (ili `~/.bashrc` datoteku za bash).

Upravljanje ključevima za implementaciju

Možete upravljati SSH ključevima na svojim serverima kada automatizirate skripte za implementaciju koristeći SSH agent za prosljeđivanje, HTTPS s OAuth tokenima, ključeve za implementaciju ili korisnike stroja.

SSH agent za prosljeđivanje

U mnogim slučajevima, posebno na početku projekta, SSH agent za prosljeđivanje je najbrža i najjednostavnija metoda za korištenje. Prosljeđivanje agenta koristi iste SSH ključeve koje koristi vaše lokalni razvojni kompjuter.

Prednosti SSH agent za prosljeđivanje

- Ne morate generirati niti pratiti nove ključeve.
- Nema upravljanja ključem; korisnici imaju ista dopuštenja na serveru kao i lokalno.
- Na serveru nisu pohranjeni ključevi, tako da u slučaju da je server ugrožen, ne morate tražiti i uklanjati ugrožene ključeve.

Nedostatci SSH agent za prosljeđivanje

- Korisnici **moraju** pristupiti SSH-u za implementaciju; ne mogu se koristiti automatizirani procesi implementacije.
- Prosljeđivanje SSH agenta može biti problematično za pokretanje za korisnike Windowsa.

Podešavanje SSH agent za prosljeđivanje

1. Uključite prosljeđivanje agenta lokalno. Za više informacija pogledajte [vodič o prosljeđivanju SSH agenta](#) .
2. Postavite svoje skripte za implementaciju da koriste prosljeđivanje agenta. Na primjer, na bash skripti, omogućavanje prosljeđivanja agenta izgledalo bi otprilike ovako: `ssh -A serverA 'bash -s' < deploy.sh`

HTTPS kloniranje s OAuth tokenima

Ako ne želite koristiti SSH ključeve, možete koristiti HTTPS s OAuth tokenima.

Prednosti HTTPS kloniranja s OAuth tokenima

- Svatko s pristupom serveru može postaviti repozitorij.
- Korisnici ne moraju mijenjati svoje lokalne SSH postavke.
- Više tokena (jedan za svakog korisnika) nije potrebno; dovoljan je jedan token po serveru.
- Token se može opozvati u bilo kojem trenutku, pretvarajući ga u biti u jednokratnu lozinku.

Nedostaci HTTPS kloniranja s OAuth tokenima

- Morate biti sigurni da ste konfigurirali svoj token s ispravnim opsegom pristupa.
- Tokeni su u biti lozinke i moraju biti zaštićeni na isti način.

Podešavanje HTTPS kloniranje s OAuth tokenima

Pogledajte [GitHub-ov vodič o stvaranju osobnog pristupnog tokena](#).

Implementacija ključeva

Možete pokrenuti projekte iz repozitorija na GitHub.com na svoj server pomoću ključa za implementaciju, koji je SSH ključ koji dopušta pristup jednom repozitoriju. GitHub prilaže javni dio ključa direktno u vaš repozitorij umjesto na osobni račun, a privatni dio ključa ostaje na vašem serveru. Za više informacija pogledajte „[Ispruka implementacija](#)“.

Ključevi za implementaciju s pristupom za pisanje mogu izvoditi iste radnje kao član firme s administratorskim pristupom ili suradnik na osobnom repozitoriju. Za više informacija pogledajte "[Uloge repozitorija u firmi](#)" i "[Nivoi dopuštenja za repozitorij osobnog računa](#)".


Prednosti ključeva za implementaciju

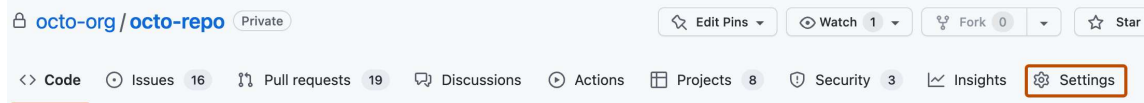
- Svatko s pristupom repozitoriju i serveru ima mogućnost implementacije projekta.
- Korisnici ne moraju mijenjati svoje lokalne SSH postavke.
- Ključevi za implementaciju su prema zadanim postavkama samo za čitanje, ali im možete dati pristup za pisanje kada ih dodajete u repozitorij.

Nedostaci ključeva za implementaciju

- Ključevi za implementaciju dopuštaju pristup samo jednom repozitoriju. Složeniji projekti mogu imati mnogo repozitorija za povlačenje na isti server.
- Ključevi za implementaciju obično nisu zaštićeni lozinkom, što ključ čini lako dostupnim ako je server ugrožen.
- Ako se korisnik koji je stvorio ključ za implementaciju ukloni iz repozitorija, ključ za implementaciju će i dalje biti aktivan jer nije vezan za određenog korisnika, već za repozitorij.

Podšavanje ključeva za implementaciju

1. [Pokrenite ssh-keygen proceduru](#) na svom serveru i zapamtite gdje ste spremili generirani javni i privatni par ključeva RSA.
2. Na GitHub.com idite na glavnu stranicu repozitorija.
3. Ispod naziva repozitorija kliknite na.  **Settings** Ako ne vidite "Settings" jahač, odaberite padajući meni, i onda kliknite na Settings.



4. Na bočnoj traci kliknite **Deploy Keys**.
5. Kliknite na **Add deploy key** tj. „Dodaj ključ za implementaciju“.
6. U polju "Title" unesite naslov.
7. U polje "Key" zalijepite svoj javni ključ.
8. Odaberite **Allow write access** (tj. Dopusti pristup zapisivanju) ako želite da ovaj ključ ima pristup pisanju u repozitorij. Ključ za implementaciju s pristupom za zapisivanje omogućava guranje implementacije u repozitorij.
9. Kliknite na Add key za dodavanje ključa.

Također možete koristiti REST API za izradu ključeva za implementaciju. Za više informacija pogledajte "[Krajnje točke REST API-ja za ključeve za implementaciju](#)".

Korištenje više repozitorija na jednom serveru

Ako koristite više repozitorija na jednom serveru, morat ćete generirati namjenski par ključeva za svaki od njih. Ne možete ponovno koristiti ključ za implementaciju za više spremišta.

U SSH konfiguracijskoj datoteci servera (obično `~/.ssh/config`), dodajte unos aliasa za svaki repozitorij. Na primjer:

```
Host github.com-repo-0
    Hostname github.com
    IdentityFile=/home/user/.ssh/repo-0_deploy_key

Host github.com-repo-1
    Hostname github.com
    IdentityFile=/home/user/.ssh/repo-1_deploy_key
```

- `Host github.com-repo-0` - Pseudonim spremišta.
- `Hostname github.com` - Konfigurira naziv glavnog računala za korištenje s aliasom.
- `IdentityFile=/home/user/.ssh/repo-0_deploy_key` - Dodjeljuje privatni ključ aliasu.

Zatim možete koristiti alias naziva hosta za interakciju sa repozitorijem koristeći SSH, koji će koristiti jedinstveni ključ za implementaciju dodijeljen tom aliasu. Na primjer:

```
git clone git@github.com-repo-1:VLASNIK/repo-1.git
```

Pristupni tokeni za instalaciju GitHub aplikacije

Ako vaš server treba pristupiti repozitoriju u jednoj ili više firmi, možete upotrijebiti GitHub aplikaciju za definiranje pristupa koji vam je potreban, a zatim generirati usko ograničene instalacijske pristupne tokene iz te GitHub aplikacije. Instalacijski pristupni tokeni mogu se proširiti na jedno ili više spremišta i mogu imati precizna dopuštenja. Na primjer, možete generirati token s pristupom samo za čitanje sadržaja repozitorija.

Budući da su GitHub aplikacije prvorazredni akter na GitHubu, tokeni za pristup instalaciji odvojeni su od bilo kojeg GitHub korisnika, što ih čini usporedivim s "uslužnim tokenima". Osim toga, tokeni pristupa instalaciji imaju namjenska ograničenja stope koja se razlikuju prema veličini firme na koju djeluju. Za više informacija pogledajte Ograničenja stope za GitHub aplikacije .

Prednosti instaliranja pristupnih tokena

- Tokeni uskog opsega s dobro definiranim skupovima dopuštenja i vremenima isteka (1 sat ili manje ako se opozovu ručno pomoću API-ja).
- Namjenska ograničenja stopa koja rastu s vašom firmom.
- Odvojeni od GitHub korisničkih identiteta, tako da ne troše nikakva licencirana mjesta.
- Nikada nije dodijeljena lozinka, pa se ne može direktno prijaviti.

Nedostaci instaliranja pristupnih tokena

- Za izradu GitHub aplikacije potrebna su dodatna podešavanja.
- Instalacijski pristupni tokeni istječu nakon 1 sata, pa ih je potrebno ponovno generirati, obično na zahtjev pomoću koda.

Postavite tokene pristupa instalaciji

1. Odredite treba li vaša GitHub aplikacija biti javna ili privatna. Ako će vaša GitHub aplikacija djelovati samo na repozitorije unutar vaše firme, vjerojatno želite da bude privatna.
2. Odredite dopuštenja koja vaša GitHub aplikacija zahtijeva, kao što je pristup samo za čitanje sadržaju repozitorija.
3. Izradite svoju GitHub aplikaciju putem stranice postavki svoje firme. Za više informacija pogledajte [Kreiranje GitHub aplikacije](#).
4. Zabilježite svoju GitHub aplikaciju `id`.
5. Generirajte i preuzmite privatni ključ svoje GitHub aplikacije i pohranite ga na sigurno. Za više informacija pogledajte [Generiranje privatnog ključa](#).
6. Instalirajte svoju GitHub aplikaciju na repozitorije prema kojima treba djelovati, po želji možete instalirati GitHub aplikaciju na sva repozitorija u vašoj firmi.
7. Identificirajte `installation_id` onu koja predstavlja vezu između vaše GitHub aplikacije i repozitorija firme kojima može pristupiti. Svaki par GitHub aplikacije i firme ima najviše jedan `installation_id`. Možete identificirati taj `installation_id` pomoću [Nabavite instalaciju firme za autentificiranu aplikaciju](#). To zahtijeva provjeru autentičnosti kao GitHub aplikacije pomoću JWT-a, za više informacija pogledajte [Provjera autentičnosti kao GitHub aplikacije](#).
8. Generirajte instalacijski pristupni token pomoću odgovarajuće REST API krajnje točke, [Kreirajte instalacijski pristupni token za aplikaciju](#). To zahtijeva provjeru autentičnosti kao GitHub

aplikacije pomoću JWT-a, za više informacija pogledajte [Autentifikacija kao GitHub aplikacija](#) i [Autentifikacija kao instalacija](#).

9. Koristite ovaj instalacijski pristupni token za interakciju sa svojim spremištima, bilo putem REST ili GraphQL API-ja, ili putem Git klijenta.

Za više informacija pogledajte "[Generiranje tokena pristupa instalaciji za GitHub aplikaciju](#)".

Strojni korisnici

Ako vaš server treba pristupiti većem broju repozitorija, možete kreirati novi račun na GitHub.com i priložiti SSH ključ koji će se koristiti isključivo za automatizaciju. Budući da ovaj račun na GitHub.com neće koristiti čovjek, naziva se strojni korisnik. Možete dodati strojnog korisnika kao [suradnika \(engl. collaborator\)](#) na osobnom repozitoriju (odobrenje pristupa za čitanje i pisanje), kao [vanjskog suradnika \(engl. outside collaborator\)](#) na repozitoriju firme (odobrenje čitanja, pisanja ili administratorskog pristupa) ili u [tim](#) s pristupom repozitoriju treba automatizirati (davanje dozvola timu).

Savjet: [Uslovi GitHub usluga](#) navode:

Računi registrirani s "botovima" ili drugim automatiziranim metodama nisu dopušteni.

To znači da ne možete automatizirati kreiranje računa. Ali ako želite stvoriti jednog strojnog korisnika za automatizaciju zadataka kao što je implementacija skripti u vašem projektu ili firmi, to je potpuno cool.

Prednosti strojnih korisnika

- Bilo tko s pristupom repozitoriju i serveru ima mogućnost implementacije projekta.
- Nijedan (ljudski) korisnik ne mora mijenjati svoje lokalne SSH postavke.
- Višestruki ključevi nisu potrebni; dovoljan je jedan po serveru.

Nedostaci korisnika stroja

- Samo firme mogu ograničiti korisnike kompjutera na pristup samo za čitanje. Osobni repozitoriji uvijek odobravaju (engl. grant) suradnicima pristup za čitanje/pisanje.
- Korisnički ključevi stroja, poput ključeva za implementaciju, obično nisu zaštićeni frazom lozinke.

Podešavanje strojnog korisnika

[Pokrenite ssh-keygen proceduru](#) na svom serveru i priložite javni ključ računu strojnog stroja.

Dajte korisničkom računu stroja pristup spremištima koja želite automatizirati. To možete učiniti dodavanjem računa kao [suradnika \(engl. collaborator\)](#), kao [vanjskog suradnika \(engl. outside collaborator\)](#) (ili u [tim](#) u firmi).

Provjera postojećih SSH ključeva

Prije nego što generirate SSH ključ, možete provjeriti imate li postojeće SSH ključeve.

O SSH ključevima

Možete koristiti SSH za izvođenje Git operacija u spremištima na GitHub.com. Za više informacija pogledajte "[Objašnjenje SSH](#)".

Ako imate postojeći SSH ključ, možete koristiti ključ za provjeru autentičnosti Git operacija preko SSH-a.

Provjera postojećih SSH ključeva

Prije nego što generirate novi SSH ključ, trebali biste provjeriti postoje li na vašem lokalnom kompjuteru postojeći ključevi.

Napomena: GitHub je poboljšao sigurnost izbacivanjem starijih, nesigurnih tipova ključeva 15. ožujka 2022.

Od tog datuma DSA ključevi (`ssh-dss`) više nisu podržani. Ne možete dodati nove DSA ključeve na svoj osobni račun na GitHub.com.

RSA ključevi (`ssh-rsa`) s `valid_after` prije 2. studenog 2021. mogu nastaviti koristiti bilo koji algoritam potpisa. RSA ključevi generirani nakon tog datuma moraju koristiti algoritam potpisa SHA-2. Neki stariji klijenti možda će trebati nadograditi kako bi mogli koristiti SHA-2 potpise.

Otvorite Git Bash .

Unesite `ls -al ~/.ssh` kako biste vidjeli postoje li postojeći SSH ključevi.

```
$ ls -al ~/.ssh
# Izlista sve datoteke u vašem.ssh direktoriju, ako oni postoje
```

Provjerite popis direktorija da vidite imate li već javni SSH ključ. Prema zadanim postavkama, nazivi datoteka podržanih javnih ključeva za GitHub su jedan od sljedećih.

- `id_rsa.pub`
- `id_ecdsa.pub`
- `id_ed25519.pub`

Savjet: Ako dobijete pogrešku da `~/.ssh` ne postoji, nemate postojeći par SSH ključeva na zadanoj lokaciji. U sljedećem koraku možete stvoriti novi par SSH ključeva.

Generirajte novi SSH ključ ili prenesite postojeći ključ.

Ako nemate podržani par javnih i privatnih ključeva ili ne želite koristiti nijedan od dostupnih, generirajte novi SSH ključ.

Ako na popisu vidite postojeći par javnih i privatnih ključeva (na primjer, `id_rsa.pub` i `id_rsa`) koje želite koristiti za povezivanje s GitHubom, možete dodati ključ u `ssh-agent`.

Za više informacija o generiranju novog SSH ključa ili dodavanju postojećeg ključa ssh-agentu, pogledajte "[Generiranje novog SSH ključa i njegovo dodavanje u ssh-agent](#)".

Generiranje novog SSH ključa i njegovo dodavanje u ssh-agent

SSH šifre ključeva

Možete pristupati i zapisivati podatke u repozitorije na GitHub.com koristeći SSH (Secure Shell Protocol). Kada se povezujete pomoću SSH-a, autentificirate se pomoću datoteke privatnog ključa na vašem lokalnom stroju. Za više informacija pogledajte "O SSH-u".

Kada generirate SSH ključ, možete dodati šifru za dodatnu zaštitu ključa. Kad god koristite ključ, morate unijeti frazu lozinke. Ako vaš ključ ima frazu lozinke i ne želite unijeti frazu lozinke svaki put kada koristite ključ, možete dodati svoj ključ SSH agentu. SSH agent upravlja vašim SSH ključevima i pamti vašu frazu lozinke.

Ako već nemate SSH ključ, morate generirati novi SSH ključ koji ćete koristiti za provjeru autentifikacije. Ako niste sigurni imate li već SSH ključ, možete provjeriti postojeće ključeve. Za više informacija pogledajte "Provjera postojećih SSH ključeva".

Ako želite koristiti hardverski sigurnosni ključ za autentifikaciju na GitHubu, morate generirati novi SSH ključ za vaš hardverski sigurnosni ključ. Morate povezati svoj hardverski sigurnosni ključ s kompjuterom kada autentificirate s parom ključeva. Za više informacija pogledajte napomene o OpenSSH 8.2.

Generiranje novog SSH ključa

Možete generirati novi SSH ključ na vašem lokalnom kompjuteru. Nakon što generirate ključ, možete dodati javni ključ svom računu na GitHub.com kako biste omogućili autentifikaciju za Git operacije preko SSH-a.

Napomena: GitHub je poboljšao sigurnost izbacivanjem starijih, nesigurnih tipova ključeva 15. ožujka 2022.

Od tog datuma DSA ključevi (`ssh-dss`) više nisu podržani. Ne možete dodati nove DSA ključeve na svoj osobni račun na GitHub.com.

RSA ključevi (`ssh-rsa`) s `valid_after` prije 2. studenog 2021. mogu nastaviti koristiti bilo koji algoritam potpisa. RSA ključevi generirani nakon tog datuma moraju koristiti algoritam potpisa SHA-2. Neki stariji klijenti možda će trebati nadograditi kako bi mogli koristiti SHA-2 potpise.

1. Otvorite **Git Bash**.
2. Zalijepite tekst ispod, zamijenivši e-poštu korištenu u primjeru svojom GitHub adresom e-pošte.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Napomena: Ako koristite naslijeđeni sistem koji ne podržava algoritam Ed25519, koristite:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Ovo stvara novi SSH ključ, koristeći dostavljenju e-poštu kao labelu.

```
> Generating public/private ALGORITHM key pair.
```

Kada se od vas zatraži "Unesite datoteku u koju želite spremiti ključ" tj. "Enter a file in which to save the key", možete pritisnuti Enter da biste prihvatili zadanu lokaciju datoteke. Primijetite da ako ste prethodno izradili SSH ključeve, `ssh-keygen` može od vas tražiti da ponovno napišete drugi ključ, u kojem slučaju preporučujemo stvaranje prilagođenog naziva SSH ključa. Da biste to učinili, upišite zadanu lokaciju datoteke i zamijenite `id_ALGORITHM` svojim prilagođenim nazivom ključa.

3. Na upit upišite sigurnu šifru. Za više informacija, pogledajte "Rad sa šiframa SSH ključa".

```
> Enter passphrase (empty for no passphrase): [Type a passphrase]
> Enter same passphrase again: [Type passphrase again]
```

Dodavanje vašeg SSH ključa ssh-agentu

Prije dodavanja novog SSH ključa ssh-agentu za upravljanje vašim ključevima, trebali ste provjeriti postojeće SSH ključeve i generirati novi SSH ključ.

Ako imate instaliran [GitHub Desktop](#), možete ga koristiti za kloniranje repozitorija i ne baviti se SSH ključevima.

1. U novom PowerShell prozoru s administratorskim ovlastima, omogućite da radi li ssh-agent. Možete upotrijebiti upute za "Automatsko pokretanje ssh-agenta" u "Rad s lozinkama ključa SSH" ili ga pokrenuti ručno:

```
# start the ssh-agent in the background
Get-Service -Name ssh-agent | Set-Service -StartupType Manual
Start-Service ssh-agent
```



2. U prozoru terminala bez dodatnih ovlasti dodajte svoj SSH privatni ključ ssh-agentu. Ako ste kreirali svoj ključ s drugim imenom ili ako dodajete postojeći ključ koji ima drugačiji naziv, zamijenite `id_ed25519` u naredbi s nazivom vaše datoteke privatnog ključa.

```
ssh-add c:/Users/JA/.ssh/id_ed25519
```

3. Dodajte SSH javni ključ vašem računu na GitHub-u. Za više informacija, pogledajte „Dodavanje novog SSH ključa vašem GitHub računu“.

Generiranje novog SSH ključa za hardverski sigurnosni ključ

Ako koristite macOS ili Linux, možda ćete morati ažurirati svoj SSH klijent ili instalirati novi SSH klijent prije generiranja novog SSH ključa. Za više informacija pogledajte "Pogreška: Nepoznata vrsta ključa".

1. Umetnite hardverski sigurnosni ključ u svoj kompjuter.
2. Otvorite Git Bash.
3. Zalijepite tekst ispod, zamijenite adresu e-pošte u primjeru sa adresom e-pošte povezanom s vašim računom na GitHubu.

```
ssh-keygen -t ed25519-sk -C "your_email@example.com"
```

Napomena: Ako naredba ne uspije i dobijete grešku `invalid format` ili `feature not supported`, možda koristite hardverski sigurnosni ključ koji ne podržava algoritam Ed25519. Umjesto toga unesite sljedeću naredbu.

```
ssh-keygen -t ecdsa-sk -C "vaš_email@primjer.com"
```

4. Kada se to od vas zatraži, dodirnite dugme na hardverskom sigurnosnom ključu.
5. Kada se od vas zatraži "Enter a file in which to save the key" tj. da unesite datoteku u koju želite spremiti ključ, pritisnite Enter da prihvatite podrazumijevanu lokaciju datoteke.

```
> Enter a file in which to save the key  
(c:\Users\YOU\.ssh\id_ed25519_sk): [Press enter]
```

6. Kada se od vas zatraži da upišete frazu lozinke, pritisnite Enter.

```
> Enter passphrase (empty for no passphrase): [Type a passphrase]  
> Enter same passphrase again: [Type passphrase again]
```

7. Dodajte SSH javni ključ svom računu na GitHubu. Za više informacija pogledajte "Dodavanje novog SSH ključa vašem GitHub računu."

Dodavanje novog SSH ključa vašem GitHub računu

Da biste konfigurirali svoj račun na GitHub.com za korištenje vašeg novog (ili postojećeg) SSH ključa, također ćete morati dodati ključ na svoj račun.

O dodavanju SSH ključeva vašem računu

Možete pristupiti i zapisivati podatke u repozitorije na GitHub.com koristeći SSH (Secure Shell Protocol). Kada se povezujete putem SSH-a, autentificirate se pomoću datoteke privatnog ključa na vašem lokalnom kompjuteru. Za više informacija pogledajte "[Objašnjenje SSH](#)".

Također možete koristiti SSH za potpisivanje commitova i tagova. Za više informacija o potpisivanju obveza, pogledajte "[O provjeri commitanja potpisa](#)".

Nakon što generirate par SSH ključeva, morate dodati javni ključ na GitHub.com kako biste omogućili SSH pristup za svoj račun.

Preduvjeti

Prije dodavanja novog SSH ključa na vaš račun na GitHub.com, izvršite sljedeće korake.

1. Provjerite postojeće SSH ključeve. Za više informacija pogledajte poglavlje "[Provjera postojećih SSH ključeva](#)".

2. Generirajte novi SSH ključ i dodajte ga SSH agentu vašeg stroja. Za više informacija pogledajte poglavlje "[Generiranje novog SSH ključa i njegovo dodavanje u ssh-agent](#)".

Dodavanje novog SSH ključa vašem računu

Možete dodati SSH ključ i koristiti ga za autentifikaciju ili potpisivanje, ili oboje. Ako želite koristiti isti SSH ključ i za autentifikaciju i za potpisivanje, trebate ga učitati dva puta.

Nakon dodavanja novog SSH autentifikacijskog ključa na vaš račun na GitHub.com, možete ponovno konfigurirati sva lokalna spremišta za korištenje SSH-a. Za više informacija, pogledajte "[Upravljanje udaljenim repozitorijima](#)".

Napomena: GitHub je poboljšao sigurnost izbacivanjem starijih, nesigurnih tipova ključeva 15. ožujka 2022.

Od tog datuma DSA ključevi (`ssh-dss`) više nisu podržani. Ne možete dodati nove DSA ključeve na svoj osobni račun na GitHub.com.

RSA ključevi (`ssh-rsa`) s `valid_after` prije 2. studenog 2021. mogu nastaviti koristiti bilo koji algoritam potpisa. RSA ključevi generirani nakon tog datuma moraju koristiti algoritam potpisa SHA-2. Neki stariji klijenti možda će trebati nadograditi kako bi mogli koristiti SHA-2 potpise.

1. Kopirajte SSH javni ključ u kontejner.

Ako vaša datoteka SSH javnog ključa ima drugačiji naziv od primjera koda, promijenite naziv datoteke tako da odgovara vašoj trenutnoj postavci. Prilikom kopiranja ključa nemojte dodavati nove retke ili razmake.

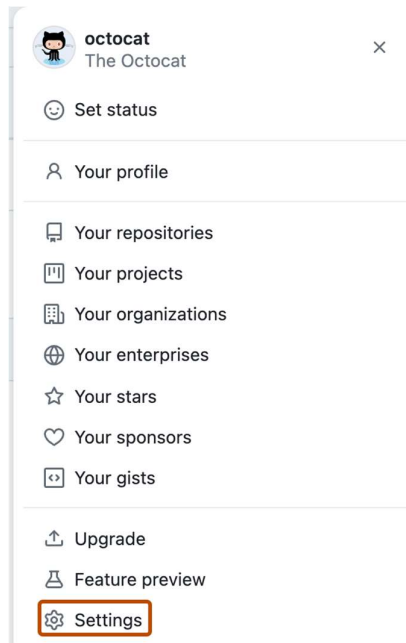
```
$ clip < ~/.ssh/id_ed25519.pub  
# Kopira sadržaj datoteke id_ed25519.pub u vaš clipboard
```

Bilješke:

- Uz Windows podsustav za Linux (WSL), možete koristiti `clip.exe`. U suprotnom, ako `clip` ne radi, možete locirati skriveni `.ssh` direktorij, otvoriti datoteku u svom omiljenom editoru i kopirati je u clipboard.
- Na novijim verzijama Windows sistema koje koriste Windows terminal ili bilo gdje drugdje koje koristi naredbeni redak PowerShell, možete primiti obavijest `ParseError` da se u ovom slučaju treba koristiti `The '<' operator is reserved for future use.` sljedeća alternativna `clip` naredbi:

```
$ cat ~/.ssh/id_ed25519.pub | clip  
# Kopira sadržaj datoteke id_ed25519.pub u vaš clipboard
```

2. U gornjem desnom kutu bilo koje stranice kliknite svoju profilnu sliku, zatim kliknite **Settings**.



3. U odlomku "Access" na bočnoj traci kliknite **SSH and GPG keys**.
4. Kliknite **New SSH key** ili **Add SSH key**.
5. U polju "Title" dodajte opisnu oznaku za novi ključ. Na primjer, ako koristite vlastiti laptop, ovaj ključ možete nazvati "Kućni laptop".
6. Odaberite vrstu ključa, provjeru autentičnosti ili potpisivanje. Za više informacija o potpisivanju obveza, pogledajte "[O provjeri commitanja potpisa](#)".
7. U polje "Key" zalijepite svoj javni ključ.
8. Kliknite na **Add SSH ključ**.
9. Ako se to od vas zatraži, potvrdite pristup svom računu na GitHubu. Za više informacija pogledajte "[Sudo mode](#)".

Testiranje vaše SSH veze

Nakon što postavite svoj SSH ključ i dodate ga na svoj račun na GitHub.com, možete testirati svoju vezu.

Prije testiranja vaše SSH veze, trebali biste imati:

- [Provjereni postojeći SSH ključevi](#)
- [Generiran je novi SSH ključ](#)
- [Dodan je novi SSH ključ vašem GitHub računu](#)

Kada testirate svoju vezu, morat ćete potvrditi autentičnost ove radnje pomoću svoje lozinke, koja je zaporka SSH ključa koju ste ranije izradili. Za više informacija o radu sa šiframa ključa SSH, pogledajte "[Rad sa SSH ključevima frazama lozinki](#)".

1. Otvorite Git Bash.
2. Unesite sljedeće:

```
$ ssh -T git@github.com
# Attempts to ssh to GitHub
```

Možda ćete vidjeti ovakvo upozorenje:

```
> The authenticity of host 'github.com (IP ADDRESS)' can't be established.
> ED25519 key fingerprint is
SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
> Are you sure you want to continue connecting (yes/no)?
```

3. Provjerite podudara li se otisak u poruci koju vidite s [GitHubovim otiskom javnog ključa](#). Ako ima, upišite **yes**:

```
> Hi KORISNIČKOIME! You've successfully authenticated, but GitHub does not
provide shell access.
```

Napomena: Daljinska naredba bi trebala izaći s kodom 1.

4. Provjerite sadrži li nastala poruka vaše korisničko ime. Ako primite poruku "dopuštenje odbijeno", pogledajte "[Error: Permission denied \(publickey\)](#)".

Rad sa frazama lozinki SSH ključa

Možete osigurati svoje SSH ključeve i konfigurirati agenta za provjeru autentičnosti tako da ne morate ponovno unositi fraze lozinki svaki put kada koristite svoje SSH ključeve.

O frazama lozinki za SSH ključeve

Sa SSH ključevima, ako netko dobije pristup vašem kompjuteru, napadač može dobiti pristup svakom sistemu koji koristi taj ključ. Da biste dodali dodatni sloj sigurnosti, možete dodati frazu lozinke svom SSH ključu. Kako biste izbjegli unos fraze lozinke pri svakom povezivanju, možete sigurno spremiti frazu lozinke u SSH agent.

Dodavanje ili promjena fraze lozinke

Možete promijeniti frazu lozinke za postojeći privatni ključ bez ponovnog generiranja para ključeva upisivanjem sljedeće naredbe:

```
$ ssh-keygen -p -f ~/.ssh/id_ed25519
> Enter old passphrase: [Type old passphrase]
> Key has comment 'your_email@example.com'
> Enter new passphrase (empty for no passphrase): [Type new passphrase]
> Enter same passphrase again: [Repeat the new passphrase]
> Your identification has been saved with the new passphrase.
```

Ako vaš ključ već ima frazu lozinke, od vas će se tražiti da je unesete prije nego što je možete promijeniti u novu frazu lozinke.

Automatsko pokretanje ssh-agent na Gitu za Windows

Možete se pokrenuti `ssh-agent` automatski kada otvorite bash ili Git shell. Kopirajte sljedeće redove i zalijepite ih u svoj `~/.profile` ili `~/.bashrc` datoteku u Git shellu:

```
env=~/.ssh/agent.env

agent_load_env () { test -f "$env" && . "$env" >| /dev/null ; }

agent_start () {
    (umask 077; ssh-agent >| "$env")
    . "$env" >| /dev/null ; }

agent_load_env

# agent_run_state: 0=agent running w/ key; 1=agent w/o key; 2=agent not running
agent_run_state=$(ssh-add -l >| /dev/null 2>&1; echo $?)

if [ ! "$SSH_AUTH_SOCK" ] || [ $agent_run_state = 2 ]; then
    agent_start
    ssh-add
elif [ "$SSH_AUTH_SOCK" ] && [ $agent_run_state = 1 ]; then
    ssh-add
fi

unset env
```

Ako vaš privatni ključ nije pohranjen na jednoj od zadanih lokacija (kao što je `~/.ssh/id_rsa`), morat ćete reći svom SSH autentifikacijskom agentu gdje ga pronaći. Da biste dodali svoj ključ ssh-agentu, upišite `ssh-add ~/putanja/do/mog_ključa`. Za više informacija pogledajte „[Generiranje novog SSH ključa i njegovo dodavanje u ssh-agent](#)“

Savjet: Ako želite da `ssh-agent` zaboravi svoj ključ nakon nekog vremena, možete ga konfigurirati tako da pokrenete `ssh-add -t <seconds>`.

Sada, kada prvi put pokrenete Git Bash, od vas se traži vaša fraza lozinke:

```
> Initializing new SSH agent...
> succeeded
> Enter passphrase for /c/Users/YOU/.ssh/id_rsa:
> Identity added: /c/Users/YOU/.ssh/id_rsa (/c/Users/YOU/.ssh/id_rsa)
> Welcome to Git (version 1.6.0.2-preview20080923)
>
> Run 'git help git' to display the help index.
> Run 'git help <command>' to display help for specific commands.
```

Proces `ssh-agent` će se nastaviti izvoditi sve dok se ne odjavite, isključite računalo ili prekinete proces.

Knjiga o git-u:

<https://tkrajina.github.io/uvod-u-git/git.pdf>

Dokumentacija:

<https://git-scm.com/docs>

Kako generirati SSH ključ moguće je vidjeti u ovom dokumentu ili na engleskom:

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>